Automatització d'algorismes de clustering i aplicació a dades mèdiques

TREBALL DE FINAL DE GRAU

Grau en enginyeria informàtica

Pau Castro Sampol

Directors: Ricard Gavaldà i Jaume Baixeries

Septembre 2017

ÍNDEX

1. CONTEXT

1.1 Introducció

La finalitat del present treball és el desenvolupament d'una eina d'anàlisi per identificar patrons de comportament en conjunt de dades mitjançant tècniques de mineria de dades, de l'anglès *data mining*. Es treballarem amb dades mèdiques, i així posar a disposició de metges una eina per el anàlisis de les dades dels seus pacients.

El data mining és el procés de computació per trobar patrons en un conjunt gran de dades mitjançant algoritmes d'anàlisi i altres eines. A partir de l'anàlisi de les dades es poden treure un gran nombre de patrons que faciliten la presa de decisions i la millora del rendiment dels processos.

L'èmfasi en aquest treball són els algoritmes de clustering, una part del data mining. El clustering segmenta les dades en subgrups, on els integrants de cada grup són similars entre si i diferents de les dades dels altres grups.

En l'àmbit de la medicina, podem disposar d'un gran volum d'informació sobre els pacients, a la que aplicant tècniques de data mining és possible trobar patrons de comportament que faciliten arribar a conclusions en salut. Atès que les dades de salut dels pacients són altament confidencials, no poden ser transferides fora de l'entorn mèdic, i com que, en general, els metges no tenen els coneixements tècnics suficients per utilitzar les tècniques de data mining, aquests estudis no es realitzen.

L'objectiu del treball és realitzar una aplicació que automatitzi el procés de clustering, amb la finalitat que els metges tinguin més facilitats alhora de poder poder aplicar tècniques de data mining amb les dades dels seus pacients.

Aquest projecte és una part d'un projecte més gran que el duu a terme el grup d'anàlisis del registres sanitaris LARCA [7], que col·labora amb diferents institucions de salut de Catalunya. El projecte consisteix, entre altres coses, en la construcció de clústers dels pacients; el usuari finals són normalment doctors amb poc coneixements de data mining. Les eines que proporcionem als usuaris han de poder-les utilitzar sense coneixements tècnics dels algorismes de clustering.

1.2 Objectius del projecte

L'objectiu principal del projecte és el desenvolupament d'una eina orientada a automatitzar les anàlisis de clustering i que els metges puguin realitzar estudis de diferent malalties a partir de les dades mèdiques de pacient i identificar patrons de comportament amb facilitat. Per aconseguir aquest resultat, hem desenvolupat diverses funcionalitats que faciliten l'anàlisi de les dades.

Les principals funcionalitats són:

1. Desenvolupar un mòdul que llegeixi les dades i les transformi al format que més adient per aplicar els algoritmes necessaris, per a cada grup de dades.
2. Desenvolupar un mòdul de tractament de les dades per a detectar i corregir valors que siguin incorrectes. Com per exemple: manca de valors en alguns casos, valors que siguin constants, errades en la entrada de dades.
3. Aplicar diversos algoritmes de *clustering* amb diferents paràmetres.
4. Analitzar els resultats amb algoritmes que calculen la bondat dels clústers, amb l'objectiu de trobar quins són els algoritmes de clustering que donen els resultats millors.

5. Desenvolupar una interfície gràfica senzilla i intuïtiva per als metges. L'aprenentatge de la interfície ha de ser ràpid per facilitar la feina.
6. Validar els resultats de la solució amb diversos jocs de proves i treure'n conclusions.

1.3 Actors implicats

El desenvolupament d'aquets projecte requereix la participació de diferents perfils:

1.3.1 Desenvolupador

Totes les tasques de desenvolupament de l'eina programa han estat realitzades per Pau Castro Sampol, que ha comptat amb l'ajut dels tutors del treball per a la resolució de dubtes i problemes.

1.3.2 Tutors del projecte

Els tutors d'aquest projecte han estat en Ricard Gavaldà Mestre i en Jaume Baixeries Juvillà que supervisaven l'acompliment del calendari estipulat i l'assoliment dels objectius marcats en el projecte. També han resolt els dubtes i problemes que han sorgit en el decurs del desenvolupament del treball de final de grau i han guiat l'estudiant a l'hora de desenvolupar el codi.

1.3.3 Usuaris

Els usuaris són els metges o personal dels centres de salut a qui va dirigit el programa i que han plantejat els requisits en base a les seves necessitats d'anàlisi i provaran la solució en el conjunt de dades de pacients amb què treballen habitualment. En principi, es treballarà amb uns usuaris pilot amb qui ja s'ha arribat a un acord.

2. Estat de l'art

2.1 Tècniques de clustering

Les anàlisis de clustering, són una part de la mineria de dades (*Data mining*), que formen part dels camps de l'estadística i de la computació i en els que s'analitzen grans volums de dades per a trobar patrons amb l'objectiu d'extreure conclusions.

La mineria de dades es fa servir en diferents àmbits, moltes empreses utilitzen les tècniques de la mineria de dades per millorar el rendiment dels processos. Posem com a exemple el cas dels supermercats que utilitzen aquets patrons per a saber quins productes estan relacionats a l'hora de comprar. Un exemple de productes relacionats en el moment de la compra en els supermercats són les cerveses i les patates fregides: els estudis realitzat pels supermercats han arribat a la conclusió que molta gent que compra cervesa, també compra patates fregides.

El principal problema del *data mining* és que es necessiten coneixements avançats en estadística i computació la qual cosa el fa inaccessible per a molta gent.

En el nostre treball, volem que a partir d'una interfície d'usuari senzilla, intuïtiva i molt visual, els metges puguin trobar patrons de comportament per certes malalties utilitzant els algoritmes de clustering, sense necessitat de tenir uns coneixements avançats de *data mining*.

Actualment existeixen molta varietat de algorismes de clustering, però tots ells necessiten d'un anàlisi per a poder extreure conclusions. El nostre treball vol automatitzar tot aquest anàlisi per a reduir els costos de temps i personal, i mostrar els resultats de les anàlisis per que els usuaris puguin extreure conclusions sense necessitat de grans coneixements d'estadística i informàtica.

2.2 Algoritmes de *clustering*

Els algoritmes de clustering o agrupament, creen conjunts de dades d'acord amb uns criteris. Aquests criteris són, en general, la similitud o la distància. Aquesta similitud és donada, normalment, a partir de unes funcions de distància. Els grups que es formen un cop aplicat l'algorisme, generalment, comparteixen unes certes propietats que permeten extreure conclusions.

L'algorisme de clustering més conegut és el k-means. L'objectiu principal de l'algorisme és dividir les dades en *k* grups. L'algorisme crea una taula en dues dimensions, on cada mostra és un punt en la taula i l'algorisme troba k punts, que es diuen centres. Cada una de les mostres pertany al grup del centre més proper.

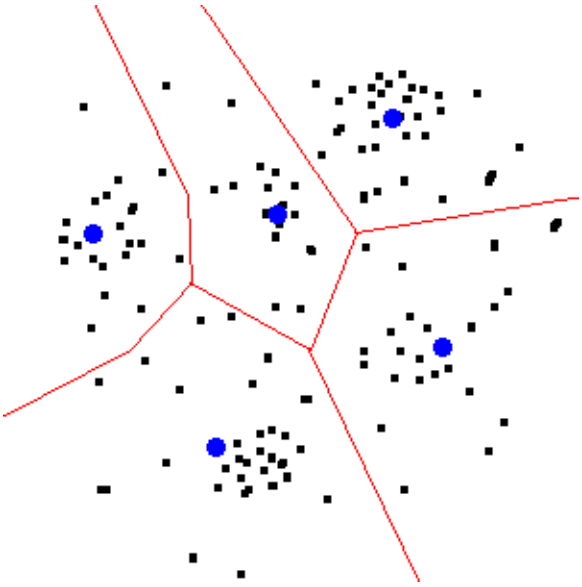

Figura 1. Exemple de k-means amb k = 5

En el nostre programa s'utilitzaran diversos algoritmes amb diversos factors (en el cas del k-means, diferents k) per a poder trobar quin de tots els resultats obtinguts agrupa millor les dades.

Cada algorisme necessita uns paràmetres alhora de realitzar els agrupaments. En general aquests són: La funció de distància (nosaltres sempre utilitzarem la distància Euclidiana) i el nombre de clústers en què volem dividir les dades.

Un cop executat cadascun dels algoritmes d'agrupació, s'avaluaran tots els grups resultants amb diferents mètodes. Un cop els algoritmes de *clustering* hagin donat diversos resultats, s'haurà de calcular la bondat dels grups. Aquesta bondat serà calculada amb uns **algorismes de validesa** per a saber quin agrupament és el més óptim. Finalment caldrà triar quin algorisme agrupa millor les dades per a mostrar els seus resultats.

L'aplicació haurà d'aplicar els següents processos en aquest ordre: prepocés, provar algorismes, avaluar resultats i mostra de resultats.

3. Definició de l'abast

3.1 Abast

Per a trobar la millor solució possible cal fixar uns passos abans de començar l'elaboració del codi.

Primer s'han d'estudiar les dades que els metges introduiran en el programa, i com aquestes poden ser tractades per a elaborar un bon algorisme de *clustering*.

Un cop analitzades les dades, s'han de transformar i modificar de tal manera que siguin molt més útils per treballar. A més, s'han d'eliminar les dades errònies o que puguin portar algun tipus de problema als algoritmes.

Un cop les dades ja han estat tractades, hem de trobar quins són els paràmetres que millor s'adapten a cada algorisme per a reduir el nombre d'execucions d'aquests. Un cop tenim totes les solucions, hem d'utilitzar els algoritmes de validació per a trobar quina solució és la més bona.

Un cop trobada la millor solució s'han de mostrar uns gràfics, molt fàcils d'entendre, amb els resultats aportats pel *clustering*.

Per últim, caldrà construir una interfície gràfica amb la qual els metges podran treballar de forma fàcil i ràpida.

3.2 Possibles obstacles

Ara analitzarem els possibles obstacles que puguin sorgir en l'elaboració del projecte, i també quines són les solucions que podríem utilitzar.

3.2.1 Errors en l'entrada de dades

Donat que l'entrada de dades la farà gent que probablement no tingui uns grans coneixements informàtics, cal prevenir la possibilitat que tinguin problemes a l'hora d'introduir les dades. Per a solucionar això, marcarem unes pautes a seguir en el moment d'entrar les dades, també es faran preguntes al usuari a través del programa per assegurar que tot va correctament.

3.2.2 Errors en les dades

Per a tractar aquestes situacions, en la majoria de casos eliminarem les dades incorrectes per a tractar-les com a nul·les. També eliminarem les dades que són o molt superiors a la majoria, o molt inferiors.

3.2.3 Temps d'execució

Com el nostre programa ha de fer servir diversos algoritmes que tenen gran cost de temps, és molt possible que el temps d'execució del programa final sigui elevat. Per sort, la nostra prioritat no és que el programa sigui molt ràpid, busquem que el programa sigui entenedor i visual més que ràpid.

3.2.4 Errors en el codi

És possible que durant el desenvolupament del projecte es produeixin molts errors en la implementació del codi. Aquets errors s'hauran d'anar controlant mitjançant proves durant el desenvolupament i test per a veure on poden estar aquets errors.

Per verificar el funcionament del programa i dels algorismes que implementarem, farem proves amb data sets.

3.4 Eines per al desenvolupament

Per a desenvolupar el programari que se'ns demana, hem utilitzat el llenguatge de programació *Python 2.7*, un llenguatge de codi obert molt fàcil d'aconseguir. L'entorn de treball serà un Mac personal de sobretaula. Utilitzaré un programari que es diu Anaconda que serveix per a programar amb Python, que ens permet editar, compilar i executar el codi de *python* de forma sencilla. Hem decidit programar en aquest llenguatge ja què la majoria de projectes en l'àmbit de les dades sanitàries al grup LARCA[7] està implementat principalment en *python*, i d'aquesta forma es podrà reutilitzar el codi d'aquest treball.

També hem utilitzat diverses llibreries en les quals molts algoritmes ja venen implementats. Aquestes llibreries són en general aquestes dues: *sklearn* i *scipy*.

Per a poder salvar el codi en línia he utilitzat el servei que ofereix Dropbox, que permet l'emmagatzemament del codi per internet. L'aplicació és privada, per tant ningú podrà veure el codi si no ho permeto jo.

4. Planificació temporal

4.1. Programa

La duració inicial del projecte era de 4 mesos aproximadament, però per problemes de temps el projecte va començar el 13 de Febrer de 2017 i acaba el 23 d' Octubre d'aquest mateix any.

Els problemes amb els temps han sigut donats per una mala organització de temps de part meva i també, la dimensió del treball que esperava inicialment no era tant gran com la real.

4.2. Pla de projecte

En aquest apartat, explicarem les diferents etapes del projecte.

4.2.1. Planificació del projecte – Fita inicial

Aquesta part del projecte és la que es va realitzar en l'assignatura de GEP i han estat actualitzades per fer la redacció d'aquesta memòria. Està dividida en les següents parts:

1. Definició de l'abast
2. Planificació temporal
3. Gestió econòmica i sostenibilitat
4. Contextualització i bibliografia

4.2.2. Anàlisi i disseny del projecte

En aquesta part del projecte, hem analitzat detalladament i especificat quin és el millor disseny final.

Durant el anàlisi hem fixat quins són els objectius que volíem assolir amb aquest projecte, hem estudiat quins són els requisits i les funcionalitats que volíem del projecte. També hem estudiat els diferents algorismes que volíem utilitzar, el seu funcionament, i les diferents forma d'avaluar la seva bondat.

Pel que fa al disseny, hem creat una petita arquitectura del software, que conté les diferents connexions que té el nostre programa final.

4.2.3. Descripció de les tasques

Per a la realització del projecte hem definit els objectius a fer durant períodes curts de temps, on hem tingut en compte les possibles desviacions o problemes que puguem tenir. També hem definit unes tasques més globals que tenen una durada més llarga, i que hem assolit mitjançant les tasques setmanals.

Ara definirem les tasques a seguir en ordre. Aquestes tasques estan dividides en tres fases, una primera fase d'anàlisi, una segona de disseny del software i l'última fase d'implementació i proves. Per la dependència entre algunes tasques, el ordre és important en el procés del projecte. Tot seguit definirem les tasques genèriques:

1. Realitzar la fita inicial.
2. Configuració de l'entorn de treball (PC, software de desenvolupador, instal·lació d'editors).
3. Creació d'una eina que llegeixi les dades i les preprocessi.

4. Aplicar integrar els algorismes de les llibreries i adaptar-los per al nostre programa.
5. Construir una eina que valori i esculli el millor resultat dels algoritmes anteriors.
6. Creació d'una interfície gràfica per al programa.
7. Executar el programa amb diversos data sets, extreure resultats i conclusions.
8. Redacció de la memòria, l'annex i la documentació.

4.3. Duració aproximada

Tasca	**Duració aproximada (h)**
Fita inicial	90
Anàlisi i disseny	30
Configuració de l'entorn	10
Creació de l'eina de llegir dades	50
Aplicar algoritmes de clustering	130
Construcció d'eina d'avaluació i selecció	100
Construcció de la interfície	80
Experimentació	20
Redacció de la memòria	40
Total	**550**

4.4. Recursos

Per a poder realitzar el projecte són necessaris diferents eines de software i de hardware. Aquí detallaré quines eines utilitzo.

4.4.1. Hardware

Les eines hardware que utilitzarem són:

- Mac Mini 2012.
- Dell XPS L502X

4.4.2. Software

Les eines software que utilitzarem són:

- Microsoft Office 2016
- Anaconda
- Python
- Dropbox versió gratuïta
- Github

5. Gestió econòmica i sostenibilitat

5.1. Gestió econòmica

5.1.1. Consideracions i comentaris

Per a poder desenvolupar aquest projecte, utilitzarem tots els elements que hem anomenat en els lliuraments anteriors, que tenen un cost. Per a poder saber el cost total del projecte, farem una estimació del cost del projecte tenint en compte recursos humans, hardware, software i costos indirectes.

Aquestes dades són les estimacions que vàrem realitzar en l'assignatura de gep.

El pressupost el calcularem a traves de les tasques representades en el diagrama de Gantt.

5.1.2. Pressupost de recursos humans

Aquest projecte serà portat a terme per una sola persona que haurà de complir amb els 4 rols presents al projecte: cap de projecte, dissenyador, enginyer de software i tester. En la següent taula mostrarem el nombre d'hores que caldran dur a terme en cada tasca i el seu valor econòmic:

Rol	Hores	Preu per hora	Preu total
Cap de projecte	100	50	5000€
Dissenyador	60	35	2.100€
Programador	290	35	10.150€
Tester	60	30	1.800€
Total	520		19.050€

5.1.3.Pressupost de hardware

Per a poder dur a terme aquets projecte, necessitem uns certs elements de hardware per a realitzar les tasques. Augmentarem aquest pressupost amb 500€ per a possibles reparacions dels dispositius. Calcularem que el temps que ho utilitzarem seran aproximadament 6 mesos.

Producte	Preu	Unitats	Vida útil	Amortització
Mac Mini	500€	1	5 anys	50€
Dell	650€	1	5 anys	65€
Total	1.150€			115€

5.1.4.Pressupost de software

Les eines de software que seran utilitzades durant les tasques del projecte són les següents:

Producte	Preu	Unitats	Vida útil	Amortització
Microsoft office Mac	270€	1	3 anys	45€
Anaconda	0€	1	--	
Python	0€	1	--	
Dropbox	0€	1	--	
Github	0€	1	--	
Total	270€			45€

5.1.5.Despeses indirectes

En tot projecte d'informàtica apareixen també algunes despeses indirectes, que en aquest cas seran el ús de l'electricitat i paper.

Producte	Preu	Unitats	Cost aproximat
Electricitat	0,11€/kWh	2.500kWh	275€

Paper	5€/pack	1 pack	5€
Total			280€

5.1.6.Pressupost total

Finalment, sumarem totes les parts del pressupost i tindrem un cost estimat del projecte.

Concepte	Cost aproximat
Recursos humans	19.050€
Hardware	115€
Software	45€€
Costos indirectes	280€
Total	19.490€

5.1.7.Control de desviacions

La principal causa de desviació que pot sorgir és la desviació temporal en alguna de les tasques del projecte. Per tal d'intentar compensar aquestes desviacions, ajustarem les tasques del diagrama de Gantt per així acabar el projecte en el temps pactat, i que les desviacions de pressupost en recursos humans siguin els mínims.

En el cas que es necessiti més pressupost perquè algun del material es trenqués, el pressupost no variaria en grans quantitats. Fent una estimació del 5% són uns 1000 €.

5.2. Àrea social

Aquest projecte té el objectiu de contribuir molt en el àmbit social, ja que vol ajudar a la medicina a millorar en el tractament i en els seus costos.

Aquest projecte pot ser de gran ajuda per a molts científics i metges que volen trobar patrons en les malalties. Per tant aquest treball vol incidir en un dels sector més importants, que és el de salut, per a facilitar la feina a metges i científics. Esperem que amb aquest projecte es pugui avançar en la construcció de guies clíniques més ajustades a la realitat de les malalties mèdiques, i que en un futur tingui una gran importància.

5.3. Àrea ambiental

En l'àmbit ambiental, aquest projecte no té un gran impacte ja que és un projecte només de software. Per tant les úniques despeses que afecten en el medi ambient són les indirectes, és a dir la llum i el paper, que per aconseguir-les, perjudiquen el medi ambient.

Per altre banda, estan la construcció dels ordinadors, que alguns dels seus elements poden tenir alguns conflictes ètics en alguns països. Alguns materials tant d'ordinadors com de mòbils han de ser portats de països estrangers i pot haver-hi certs conflictes en aquests països.

6. Introducció a la Mineria de dades

La mineria de dades (data mining en angles) és un camp de les ciències de computació i de l'estadística que vol analitzar gran conjunt de dades per a trobar-hi patrons en elles. La mineria de dades troba informació en conjunt de dades (en general aquest volum de dades és extens) i extreu uns resultats que serveixen per analitzar les dades que es tenen. Aquestes tècniques s'apliquen amb l'objectiu de trobar informació útil, que d'altre manera no podria ser trobada.

6.1. Estructura de l'anàlisi de dades

Un estudi de mineria de dades normalment consta dels següents passos.

6.1.1. Selecció i anàlisi del conjunt de dades

Primer hem de seleccionar amb quin conjunt de dades volem treballar i fer un estudi de les variables que conté. És important en aquest pas que les dades triades siguin dins d'una població definida de la qual es vol treure conclusions

6.1.2. Preparació del conjunt de dades

Aquest pas és conegut com a preprocés. En ell haurem de tractar les dades segons els estudis fets en el pas anterior. Hem de tenir present que aquest pas vol transformar les dades per a que els estudis que fem posteriorment siguin els més exitosos possibles. Si no fem un bon preprocés de les dades és més difícil que l'estudi obtingui resultats útils.

En aquest apartat hem de fer un extens estudi de les variables, és important trobar tots aquells valors atípics. En aquest pas és important fer ús de histogrames i diagrames per a la cerca dels valors atípics, ja que els algorismes de clustering es confonen molt amb aquests valors. També s'ha de de fer un estudi dels valors nuls o mancants. Els valors nuls o atípics normalment són descartats o transformats al valor mig de la variable.

6.1.3. Aplicació de les tècniques de mineria de dades

Un cop definit el conjunt de dades i hem fet el preprocés d'elles, hem de triar quina tècnica de mineria de dades volem aplicar. Hi ha dues grans famílies de tècniques en mineria de dades: supervisada i no supervisada.

1- **Tècniques supervisades:** En aquestes tècniques hi ha una variable anomenada sortida que ens diu quina hauria de ser la resposta correcta del model. Els dos tipus principals són la classificació, en què la sortida és una variable nominal, i la regressió, quan la sortida és una variable numèrica. Alguns dels models per a fer aprenentatge supervisats, la regressió lineal i logística, i els arbres de decisió.
2- **Tècniques no supervisades:** Aquestes tècniques no disposen d'una indicació de quina és la sortida correcta, en canvi tracten de trobar estructura en les dades. Les dues tècniques principals són el clustering, que agrupa els punts de les dades en grups homogenis i la cerca de patrons freqüents, que troba associacions complexes entre els atributs d'aquests exemples.

En aquest projecte, com hem dit, ens concentrarem en la tècnica no supervisada clustering.

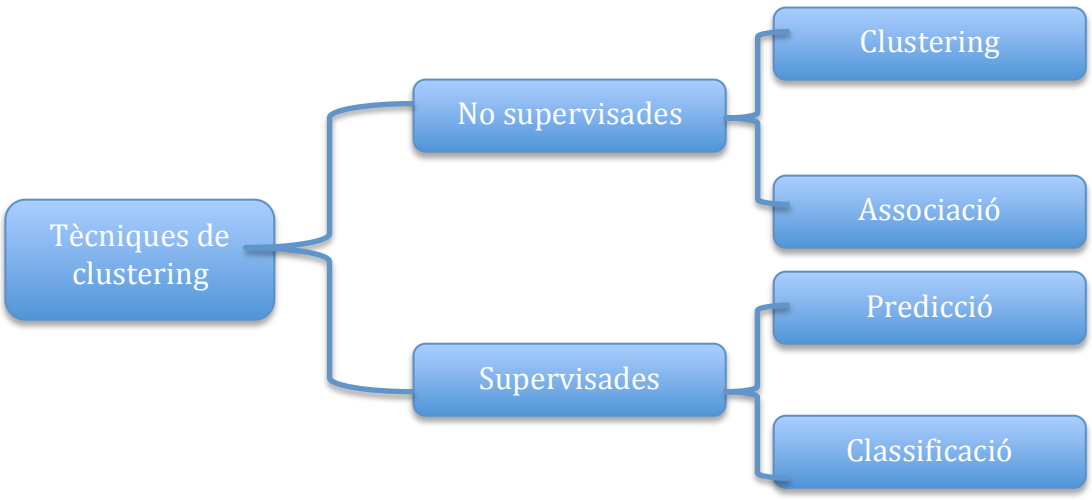

Figura 2, esquema amb els diferents tipus d'estudi de data mining.

7. Anàlisis de clustering

Les anàlisis d'agrupació, o clustering en angles, són un tipus d'anàlisi dins de la mineria de dades que agrupa vectors d'un mateix conjunt de dades d'acord a uns criteris, que en general aquests criteris són de distància. Per al càlcul dels grups s'apliquen uns algorismes que són anomenats de clustering. Per agrupar les dades, aquests algorismes utilitzen funcions de distància i similaritat. La forma més comuna de calcular les distàncies entre els vectors més utilitzat és la funció de distància euclidiana[1].

Aquest algorismes agrupen les dades en grups. El seu objectiu és que les dades d'un mateix grups siguin molt semblants, i les dades de dos grups diferents siguin el més diferents possibles.

Aquest procés no és un procés automàtic, és un procés elaborat perquè cal examinar diferents configuracions de manera exhaustiva. Una de les decisions importants que cal prendre és el nombre de grups a formar; d'entrada no sabem quants grups "hi ha" en les dades. Alguns algorismes determinen aquest número autònomament a partir de les dades. En altres casos exigeixen que el proporcioni l'usuari, com en el cas de l'algorisme més conegut, l'algorisme k-means. En el nostre treball buscarem la forma de com automatitzar aquest procés de la millor forma possible.

7.1. Utilitats del anàlisi de clustering

Ara posarem uns exemples d'ús de les anàlisis de clustering, evidentment hi ha moltes més utilitats.

---

[1] Distància euclidiana és la distància entre dos punts de R^n en el espai Euclidià R^n. Siguin P i Q dos punts de dimensió n, la distància entre els dos punts és: d(P,Q) = $\sqrt{(P_1 - Q_1)^2 + \cdots + (P_n - Q_n)^2}$

Anàlisis de les xarxes socials: En aquest àmbit s'utilitza els clustering per reconèixer grups de persones dins de grups més grans. Aquest estudis són utilitzats per moltes campanyes de màrqueting per a definir perfils de clients i dissenyar campanyes adequades a cada perfil.

Sistemes de recomanació: Segons els gustos que es pugui detectar del usuari se'l pot associar amb un grup de gent i recomanar elements nous segons grups d'usuaris. Un exemple molt clar és la plataforma www.youtube.com . Clustering és un dels diversos mètodes que es pot fer servir per a la recomanació.

En educació: Serveix per a determinar quins són els grups d'estudiants, per saber quins són els que més probablement que requereixin d'una atenció especial, o per a dissenyar un conjunt de currículums adequat a cada grup.

En els supermercats: Un exemple clar d'ús de les tècniques de clustering és els supermercats. Les utilitzen per a identificar grup de clients i crear diferents tipologies de clients. Aplicant aquestes tècniques es varen adonar que les tardes de dies en els quals hi havia partits de futbol amb gran audiència, la compra de cerveses i de patates fregides. En general els compradors eren homes adults. Per aquest motiu varen decidir posar la cervesa i les patates fregides una al costat de l'altre, aconseguint així un augment de les vendes dels dos productes.

També en els supermercats es van adonar, gràcies a un estudi de mineria de dades, que els divendres per la tarda hi havia un percentatge de venda de bolquers i de cerveses respecte la resta de dies. La raó és que pares joves tenien pensat quedar-se el cap de setmana amb els nadons a casa. Amb aquest estudi varen decidir juntar aquests dos productes els divendres per a incrementar els guanys.

En els negocis: Un nombre gran de botigues d'aliments varen fer un estudi de tots els seus clients més habituals. Amb la realització d'aquest estudi varen

veure que hi havia 5 grans grups de compradors, un exemple de comprador era aquell que preferia la compra de productes frescos. Amb aquest estudi varen plantejar estratègiques de màrqueting per a cadascun dels grups.

Dades mèdiques: Un exemple d'agrupament de dades és el que va dur a terme en Ricard Gavaldà i en Matteo Ruffini[8]. En el treball varen treballar en unes dades binaries d'alta dimensió. Amb l'anàlisi de clústers, varen dividir el data set amb el que treballaven en 5 grups. Amb aquests grups es van poder construir tractaments guiats en els pacients.
En el treball farem experiments amb dades molt similars als dels estudis [8] i

8. Problema a solucionar

Com tots sabem, la tecnologia és la ciència que resol problemes concrets, doncs aquest treball vol solucionar el problema que hem explicat abans: Clusteritzar un conjunt de dades de forma automàtica.

Per tant el que volem aconseguir és automatitzar un procés que precisa de un estudi extens i personal amb coneixements. Per aconseguir automatitzar el procés de clustering utilitzarem tècniques apreses durant aquests anys. Això comporta diversos passos:

1- Estudi dels algorismes d'agrupació.
2- Estudi del nombre de grups a dividir les dades.
3- Estudiar com valorar els resultats obtinguts per aquests algorismes.
4- Estudiar quina de les maneres d'agrupar les dades és la millor.

Amb aquestes quatre estudis esperem trobar una forma d'aconseguir el nostre principal objectiu.

9. Estudi dels algorismes d'agrupació

Per a poder fer una cerca de clustering adequat, hauríem d'utilitzar el màxim nombre d'algorismes possibles. Per problemes de temps en aquest treball hem decidit reduir el nombre d'algorismes, és per aquest motiu que solament hem utilitzat dos algorismes intentant que aquest dos siguin el més diversos possibles.

Hi han molt algorismes de clustering, entre altres estan els següents: basats en particions, jeràrquic aglomeratiu, jeràrquic divisiu, basats en densitat, basats en descomposició espectral... Per aquest treball hem triat els dos algorismes més usats i ben descrits que són: l'algorisme k-means que està basat en particions, i un mètode jeràrquic aglomeratiu.

Ara descriurem els dos algorismes amb els quals treballem en aquest treball.

9.1. Algorisme de K-means

Aquest algorisme va ser introduït per Stuart Lloyd l'any 1957. L'algorisme intenta optimitzar una funció que té complexitat NP-difícil. No obstant, és una heurística fàcil d'implementar, que funciona ràpid gairebé sempre i els seus resultats són acceptables gairebé sempre. Per aquests motius és l'algorisme de clustering més utilitzat actualment, tot i que n'hi ha que poden donar solucions de molta més qualitat.

El objectiu principal de l'algorisme és trobar k centres i dividir les dades d'entrada (direm que el nombre de les dades d'entrada és n) en k grups. Cada observació s'assignarà a aquell centre més similar a ella. Aquest nombre k és definit prèviament a l'execució de l'algorisme.

Per a triar a quin dels centres cada observació s'assembla més, la tècnica que nosaltres utilitzarem i una de les més utilitzades és la distància euclidiana. Per

tant, un cop establerts els centres cada observació pertanyerà a aquell centre més proper en quant a distància euclidiana.

El funcionament de l'algoritme és el següent:

- Inici: Es defineixen *k* centres que es poden definir de diferents maneres. La forma més típica és triar *k* observacions de forma aleatòria i seleccionar-les com a centres. L'altre es fixar *k* centres el més allunyat possible entre ells.
- El segon pas és assignar cada observació al centre més proper. Un cop format els *k* grups, és calculen els baricentres d'ells. Aquests baricentres passen a ser els nous centres.

L'algorisme repeteix el segon pas fins a que els centres no variïn.

La funció que intenta optimitzar l'algorisme és la següent:

Si tenim els punts en el espai $x_1 \dots x_n$, i els centroides d'aquests són $c_1 \dots c_k$ la funció que optimitza és:

$$\sum_{i=1}^{n} dist(x_i, c(x_i))$$

on $c(x_i) = argmin(dist(x_i, c_j) : j = 1..k)$

El cost de l'algorisme és: $O(nkdi)$, on *n* és el nombre d'observacions total, *k* el nombre de centres, *d* és el nombre de variables que tenim de cada dada i *i* és el nombre d'iteracions que fa l'algorisme.

9.2. Algorisme Hierarchy

Els algorismes de clustering aglomeratius (hierarchical clustering) agrupen les dades seguin un arbre de jerarquia de grups. Hi han dos tipus d'algorismes, els algorismes aglomeratius i els algorismes divisius:

- **Aglomeratiu:** Inicialment divideix totes les dades en clústers, de tal forma que cada observació forma un clúster. Un cop creat tots els clústers va ajuntant-los per semblança, fins a tenir un sol clúster.
- **Divisiu:** Aquest mètode fa el contrari al anterior, inicialment totes les dades pertanyen al mateix clúster i va dividint el grup segons semblança.

Cada algorisme té una complexitat diferent, l'algorisme aglomeratiu, implementat adequadament, té una complexitat (considerant que el càlcul de les distancies és constant) de $O(n^2 * \log(n))$ i el divisiu té una complexitat de $O(2^n)$. En el nostre treball esperem una entrada de dades molt gran, com podem veure en les dues formules de cost, el temps d'execució de l'algorisme serà elevat en els dos casos però l'algorisme aglomeratiu és més ràpid en entrades grans. Per aquest motiu nosaltres utilitzarem l'algorisme aglomeratiu.

En aquest tipus d'algorismes, sempre és útil l'ús dels dendrogrames. Un exemple de dendrograma el tenim en la figura 2, on podem veure un gràfic en forma d'arbre. Cada una de les arrels del arbre és una dada, per tant si tallem el gràfic per el eix horitzontal amb alçada 0, es formarien tants clústers com dades. Segons anem tallant el gràfic amb més alçada, les arrels originals es van unint formant clústers fins arribar a la part superior del arbre, on totes les dades són agrupades en un sol clúster. És molt important l'alçada de cada branca, perquè representen la distància que hi ha entre clústers.

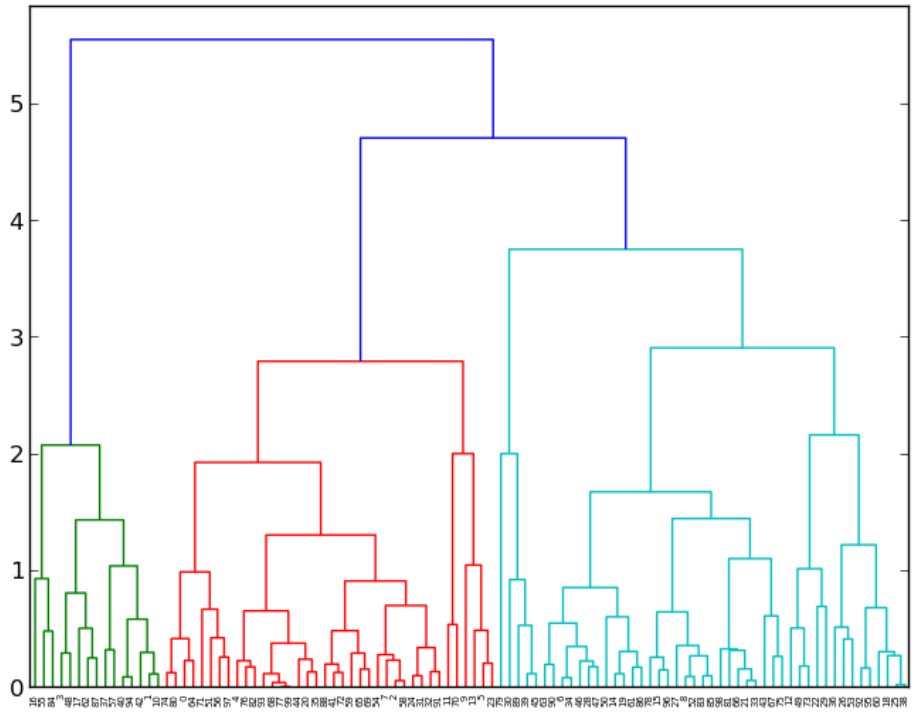

Figura 3, exemple de clustering presentat en forma de dendrograma [5].

Per a calcular la similitud entre les dades aquests algorismes poden utilitzar qualsevol formula de distància. Nosaltres utilitzarem la distància euclidiana ja que en l'algorisme k-means és la que emprem. Amb coneixements específics dels dominis es podrien construir funcions de distància que generarien clústers més útils a la pràctica.

10. Estudi del nombre de grups a dividir les dades

Saber en quants clústers s'han de dividir un conjunt de dades, és un problema molt comú en les anàlisis de clustering. Molts algorismes la definició de en quants clústers es vol dividir les dades (*k*) és necessari, en canvi en altres algorismes com en el aglomeratiu, no és necessari.

Determinar aquesta *k* no sempre és senzill. Per una part volem que es creïn grups que les dades dins del grup siguin molt semblants i dades de dos grups diferents molt diferents, però tampoc volem que el nombre de clústers sigui molt gran. Per tant, si no volem que el nombre de clústers sigui gran s'ha de penalitzar que hi hagin més clústers.

Una forma de veure-ho seria dir: "busquem aquella *k* tal que si afegim un clúster, el guany que obtenim no és prou significatiu".

Per a saber quin és el nombre òptim de clústers existeixen diversos mètodes. Nosaltres hem utilitzat dos dels mètodes més freqüentats que són l'*elbow method* [10] i el *silhouette* [11].

El que farem es aplicar cadascun dels mètodes amb els que obtindrem dos valors (el ideal seria que amb els dos mètodes obtinguéssim el mateix valor) que els direm *k* i *s* (*k* el obtingut amb l'*elbow method* i *s* el obtingut amb el *silhouette*). Un cop obtinguts aquests dos valors buscarem el clustering òptim amb el nombre de clústers que estigui compres entre aquests dos.

Ara explicarem el funcionament dels dos mètodes:

10.1. Elbow method

Aquest mètode tria el millor nombre de clústers en funció de la variància dins dels clústers que hi ha respecte del nombre de clústers en què dividim les dades. Per a calcular aquesta variància s'han de sumar totes les distancies euclidianes dos a dos entre els vectors d'un mateix clúster.

Per l'explicació del mètode usarem un exemple molt clar[15]. En aquest exemple utilitzen el diagrama de punts de la figura 3. Simplement veient la imatge veiem com es poden formar tres grups clarament. Ara anem a analitzar aquest conjunt de dades amb *l'elbow method*. Per al seu anàlisi és molt útil el ús d'un diagrama que mostri la variància dins dels clústers respecte del nombre de clústers. Aquest diagrama el trobem a la figura 4.

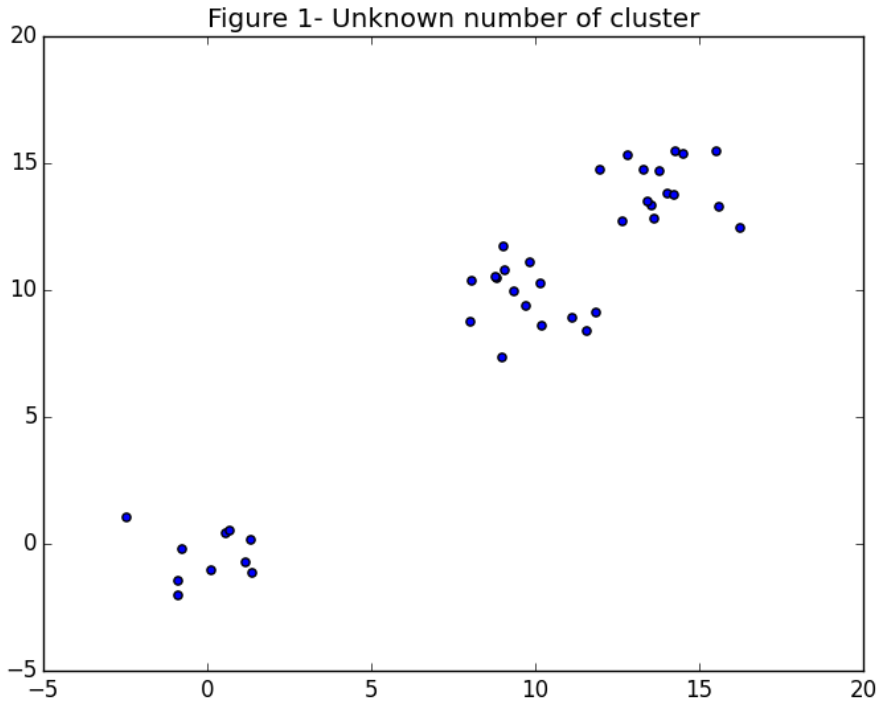

Figura 4, exemple de gràfic de dispersió de dades[3].

Com podem veure en la figura 4, si agrupem les dades en un sol clúster tenim una suma de les variàncies molt elevada comparada amb els altres nombres de clústers. També podem observar que a partir d'un cert nombre de clústers, en aquest exemple 3, per cada clúster que afegim la variància no es redueix tant. Si observem el dibuix de la gràfica en el punt que té coma nombre de

clústers 3, veiem com es forma un colze (*elbow* en angles). Utilitzant aquest mètode aquest seria el nombre de clústers a elegir.

En altres casos podríem trobar-nos que aquest colze no és tant fàcil de identificar, i segurament no tothom triaria el mateix punt ja que, pel que hem vist, no té una definició matemàtica generalment acceptada. Per tant, nosaltres n'haurem de triar una de les proposades, o proposar-ne una de nova.

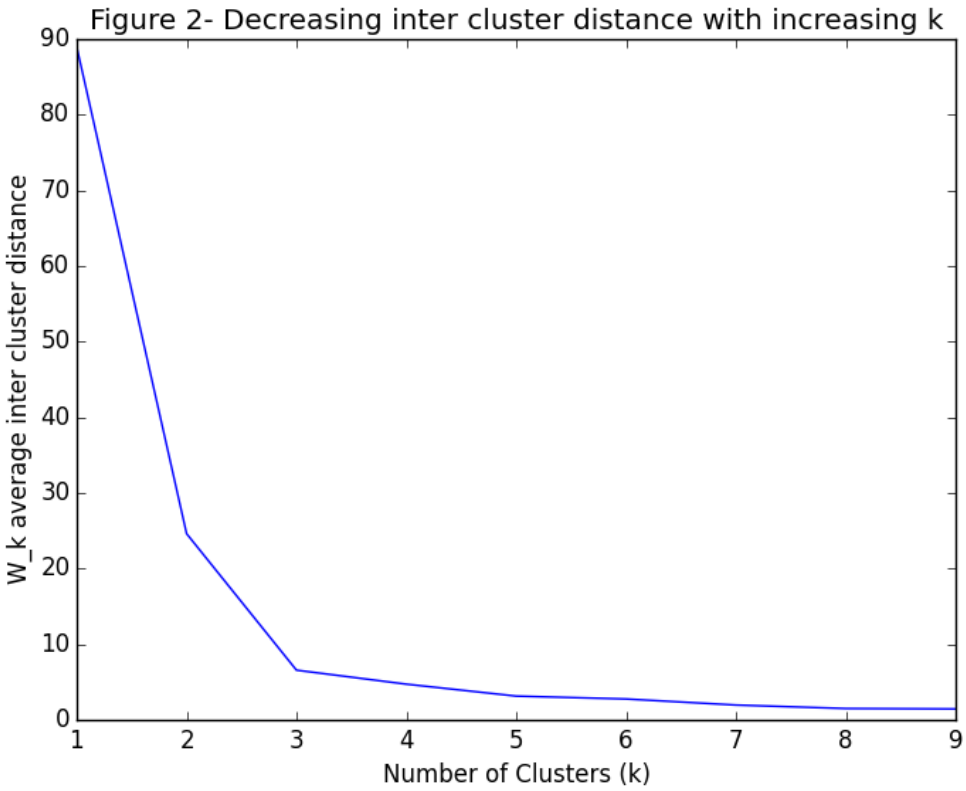

Figura 5, exemple de gràfic de distància dels elements del mateix clúster[4].

10.1.1.Implementació

La noció de colze no és del tot única i s'han fet diverses propostes. Les tres propostes que hem trobat més interesants són:

- El punt de la corba amb la 2a derivada més gran. És a dir, trobar el punt de curvatura més alt.

- El punt de la corba on la tangent és paral·lela a la línia que uneix el punt inicial i final de la corba.
- El punt més llunya a la línia que uneix el punt inicial i final de la corba.

Finalment ens hem decantat utilitzar el tercer mètode, tot i que també utilitzarem idees de les altres dues. Buscarem el punt més llunya a la diagonal, que coincidirà amb aquell punt que té com a tangent una recta paral·lela a la diagonal.

Primer de tot hem decidit delimitar el nombre de clústers entre 2 i $\sqrt{n}$ (el nombre de dades d'entrada és *n*), perquè com treballarem amb entrades molt grans, i el nombre de clústers amb els quals volem dividir les dades no ha de ser molt gran, hem trobat convenient reduir el nombre màxim de clústers.

Com tots els diagrames amb els que ens esperem trobar fan forma de L, per a trobar el colze hem decidit que buscaríem aquell punt que tingui la pendent igual que el de la diagonal que connecta el nombre màxim de clúster amb el nombre mínim.

Com la corba té pendent sempre és negatiu i creixent, i el punt que busquem és únic, eficient la cerca d'aquest punt farem servir una cerca dicotòmica.

Si tenim una funció decreixent i continua en forma de L, on el primer punt és el màxim de la funció i el últim punt és el mínim, la derivada discreta d'aquesta funció serà una funció creixent i continua sempre menor a 0, on per un punt j de la funció, $f'(j) > f'(j+1)$, sent $f(i) = f(i+1) - f(i)$. Direm que la pendent que busquem és *i* (pendent entre $f(2)$ i $f(\sqrt{n})$). Si comencem calculant el pendent en un punt *x,* punt situat a la meitat dels dos extrems, es poden donar dos casos:

- Si $f'(x) > i$ el punt que estem buscant estarà entre *x* i el extrem dret.
- En canvi si $f'(x) < i$ el punt que estem buscant estarà entre l'extrem esquerre i *x.*

Si ara *x* passa a ser el nou centre entre els dos extrems i repetim les condicions anteriors, i repetim això fins que els dos extrems siguin el mateix punt, aquest serà el punt que estàvem buscant.

Aquest mètode acaba tenint un cost de $O(k)$, sent *k* el nombre de clústers provats, que com a màxim serà $\sqrt{n}$. Per tant podem dir que el mètode tindrà un cost total de $O(n)$.

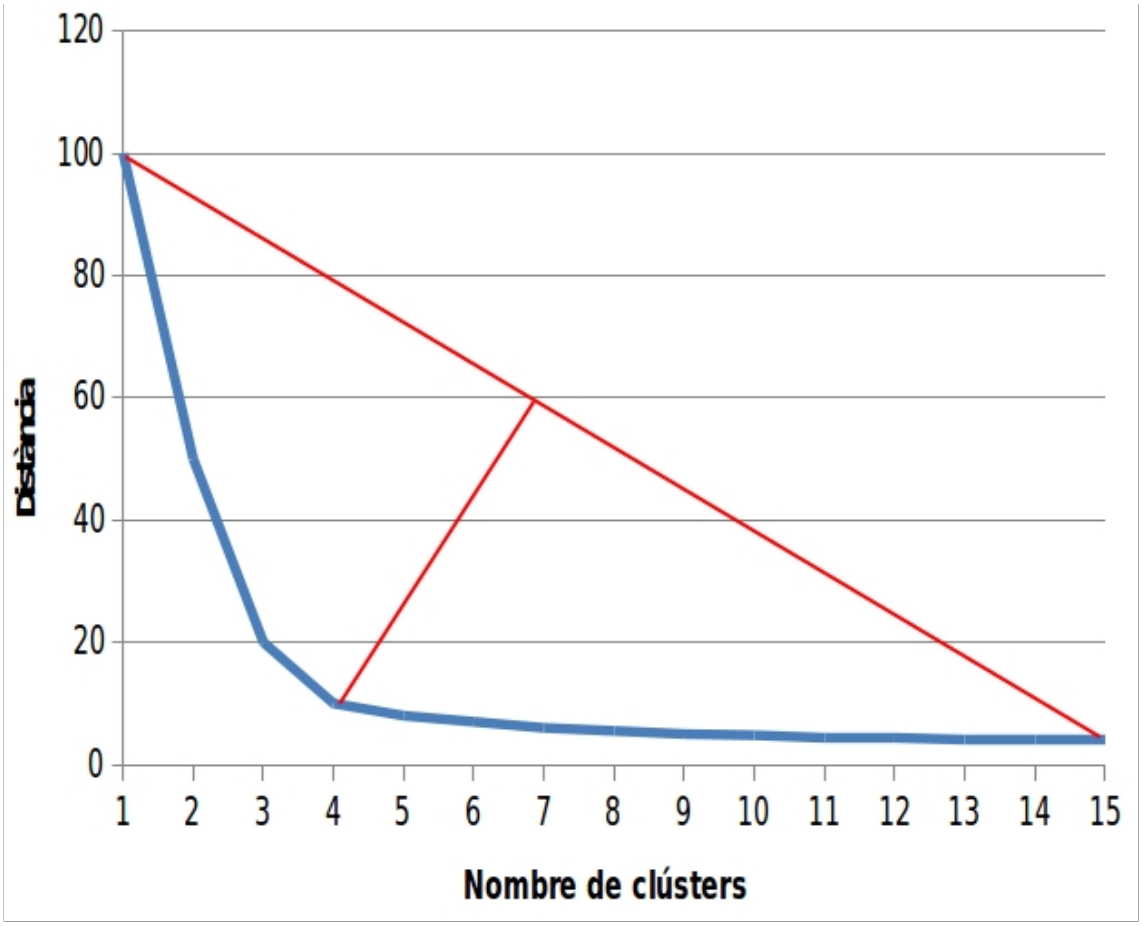

Figura 6, exemple de gràfic de l'elbow method.

10.2. Silhouette

Aquest mètode si que té una definició matemàtica ben acceptada, que serveix per calcular la consistència dels clusters. Aquesta funció el que fa és comparar la similitud de cada observació amb les observacions dels altres clústers. El

resultat d'aquesta funció és un índex entre -1 i 1, quan més alt és aquest valor, millor és l'agrupament de les dades.

Per aquest mètode també podem ajudar-nos amb un diagrama com el de la figura 5. En aquest cas també hem de seleccionar aquell nombre de clústers tal que afegint un altre clúster no observem una gran millora.

Per calcular la similitud de les dades, hem utilitzat la distància euclidiana com hem vingut utilitzant anteriorment.

Formula de l'índex:

Primer de tot hem de dividir les dades en *k* clústers. Un cop dividides, anomenarem a una observació qualsevol *i*. Definirem com $a(i)$ la distància mitjana del punt *i* a totes les altres observacions del clúster al qual pertany. Definirem com a $b(i)$ a la distància mitjana a tots els punts del clúster més proper (entenem al més proper com aquell clúster amb el centre més proper), sense tenir en compte el propi clúster d'*i.*

Un cop calculat aquest dos valors, apliquem la formula:

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

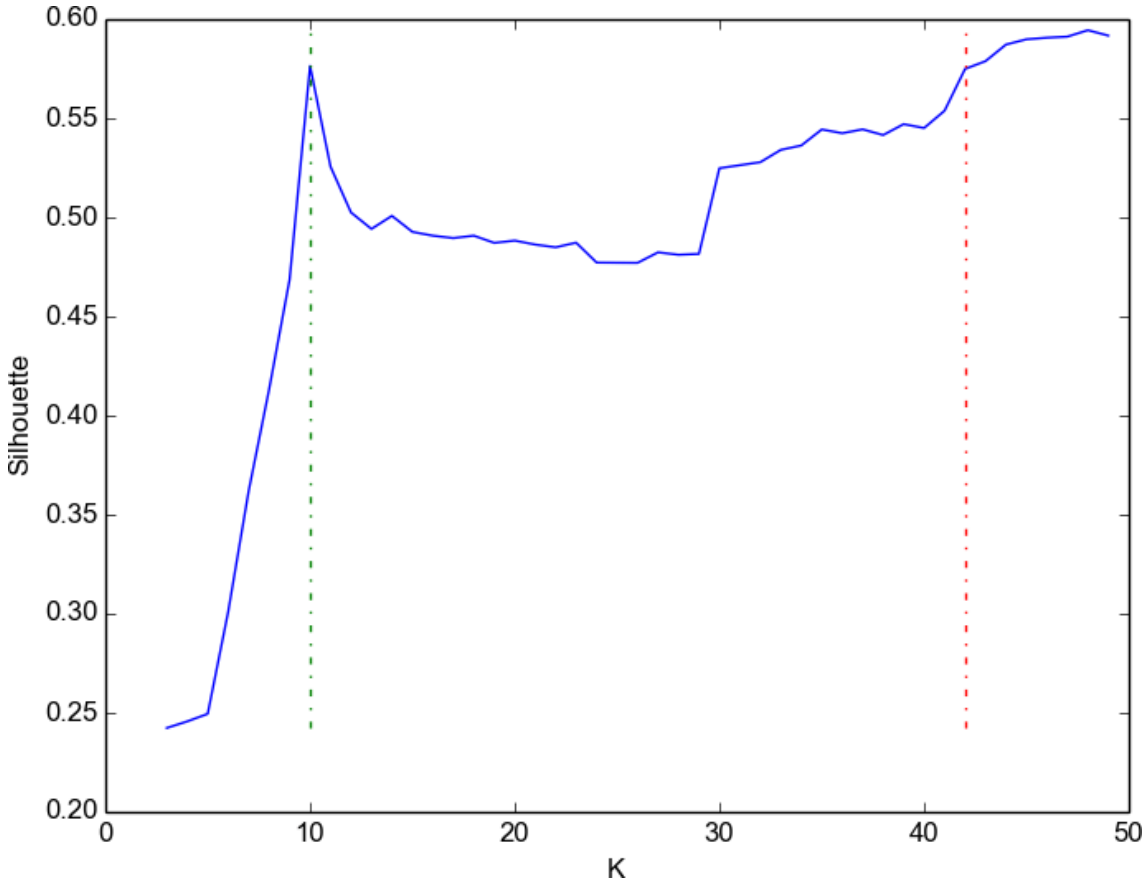

Figura 7, exemple de gràfic dels valors de Silhouette respecte el nombre de clústers [6].

Una de les conclusions que podem extreure al analitzar la formula és: si el valor $a(i)$ és molt inferior al de $b(i)$, obtindrem un valor proper a 1, que significaria una bona divisió per a aquesta observació. Si pel contrari $b(i) \ll a(i)$, obtindrem un valor de l'índex proper a 0 o negatiu, que diria que l'observació no està molt ben agrupada.

Per a obtenir el valor general en el conjunt de dades, hem de calcular aquest valor per a cada observació i fer la mitjana de tots els valors obtinguts.

10.2.1.Implementació

Per al càlcul d'aquest l'índex d'un agrupament de dades hem utilitzat la llibreria per python sklearn. Per a no tenir que calcular tots els valors de silhouette, només hem calculat aquest valors amb valors de *k* que s'obtenen amb aquesta funció: $f(x) = 2^{x-1}$.

Un cop calculat aquests valors, esperem trobar-nos sempre amb una funció semblant a la de la figura 5. Un cop analitzats tots els valors obtinguts, continuarem aplicant la cerca dicotòmica entre els dos punts següents: el primer punt serà aquell de la gràfica tal que els valor de silhouette calculats següents (les dues *k* superiors a aquest punt), no millorin més d'un 5% l'actual. L'extrem esquerre serà el valor de *k* adjunt en el punt anterior però més petit. Un cop tenim aquest dos nous extrems tornarem a fer la cerca dicotòmica entre aquests dos valors, i així successivament fins que els dos extrems siguin adjacents (*k* i *k*+1).

El cost d'aquest mètode és bastant més elevat que el de l'elbow, el seu cost és $O(n^2)$.

11. Estudi de com valorar els resultat obtinguts pels algorismes

Un cop aplicat hem aplicat els algorismes i hem delimitat el rang del nombre de clústers, tindrem quatre valors: k_{kmeans} (obtinguda amb l'algorisme k-means i el mètode elbow), s_{kmeans} (obtinguda amb l'algorisme k-means i el mètode silhouette), $k_{hierarchy}$ (obtinguda amb l'algorisme hierarchy i el mètode elbow) i $s_{hierarchy}$ (obtinguda amb l'algorisme hierarchy i el mètode silhouette). Tots els clusterings obtinguts per l'algorisme k-means amb k entre k_{kmeans} i s_{kmeans} són en principi candidats a millor clustering, el mateix passa anàlogament amb els valors obtinguts amb silhouette.

Per tant tindrem fins a $|k_{kmeans} - s_{kmeans}| + |k_{hierarchy} - s_{hierarchy}| + 2$ diferents clusterings que són candidats a ser la solució final. Un cop tenim tots els clusterings hem de triar quin d'aquests és el millor.

Hi han diverses tècniques que valoren la bondat dels agrupaments, però no podem dir que cap sigui millor ni pitjor que les altres. És per això que hem decidit utilitzar les tres tècniques més utilitzades que són: Silhouette [11], Dunn Index[9] i Davies-Bouldin[12].

11.1. Silhouette

Aquest és el mateix índex que utilitzem per a la tria del nombre de clústers. És un índex útil per a valorar com de bo és un agrupament ja que té en compte quan de compacte són els clústers respecte els altres.

Cal recordar que els valors del silhouete estan entre -1 i 1.

11.2. Dunn Index

Igual que els altres índexs, aquest índex valora com de compacte són els clústers, havent-hi poca diferencia entre els elements d'un mateix clúster, i que els clústers estiguin ben separats.

En aquest índex, com més alt és el valor que obtinguem millor és l'agrupament.

Ara mostrarem la formula que aplica aquest índex:

Primer calcularem per cada clúster el seu diàmetre, que definim com a la distància màxima entre dos dels integrants d'un mateix clúster (C_i).

$$\Delta_i = \max_{x,y \in Ci} d(x,y)$$

Després definirem la distància entre clústers, com a la distància euclidiana entre els seus centres. La denotarem com: $\eth(C_i, C_j)$.

Amb aquestes dues definicions, i tenint *m* clústers, la formula del Dunn Index és:

$$DI_m = \frac{\min_{1 \le i < j \le m} \eth(C_i, C_j)}{\max_{1 \le k \le m} \Delta_k}$$

11.3. Davies-Bouldin

Aquest índex fa uns càlculs semblants al Dunn. En aquest cas es compara la distància mitjana dels punts d'un clúster al seu centre amb la distància entre centres.

A diferencia de l'anterior, com més petit sigui el valor que obtenim millor serà aquest agrupament de les dades.

Abans de mostrar la formula complerta, definirem dos valors. Sigui *m* el número de clústers i *x* un clúster, definirem com σ_x com la distància mitja de tots elements del clúster *x*, al seu centre. També, sent C_i i C_j dos centres de clústers, definirem com $d(C_i, C_j)$ la distància entre els centres dels clústers *i* i *j*.

La formula del Davies-Bouldin és:

$$DB = \frac{1}{n}\sum_{i=1}^{n} \max_{i \neq j}\left(\frac{\sigma_i + \sigma_j}{d(C_i, C_j)}\right)$$

11.4. Tria del millor clustering

Un cop ja hem calculat tots els índexs per a totes les diferents agrupacions, construïm una matriu amb *n* (nombre de diferents agrupacions) files i 3 columnes. Els índexs estaran distribuïts per files en el ordre que els hem descrit abans (Silhouette, Dunn, Davies-Bouldin). Per a saber quin és el millor clustering fem servir aquesta formula:

$$BC = \max_{1 \leq i \leq n}\left(\sum_{j=1}^{2} \frac{mat_{ij}}{\max\left(col(j)\right)} + 1 - \frac{\min\left(col(3)\right)}{mat_{i3}}\right)$$

La fila que obtingui el valor més alt amb aquesta formula serà la que seleccionarem com a millor agrupament.

12. Desenvolupament del software

Ara explicarem com hem fet el desenvolupament d'aquest software.

12.1. Diagrama UML

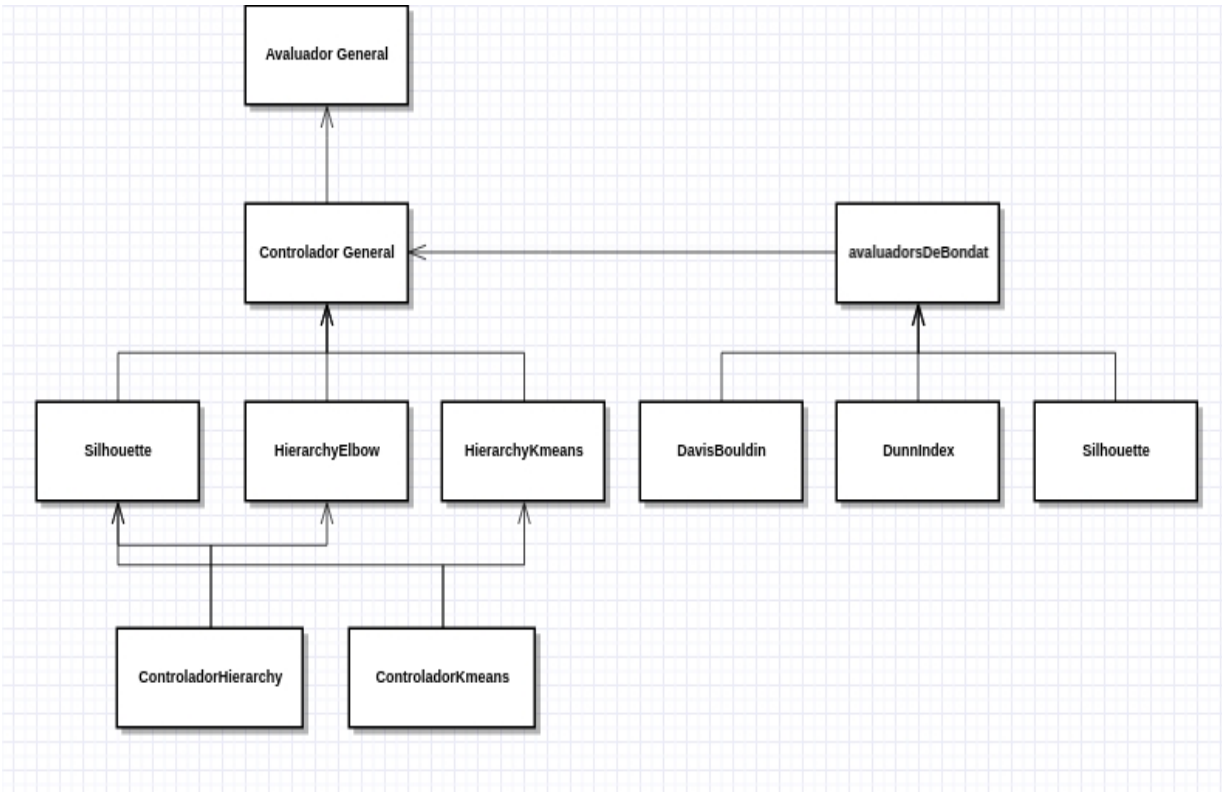

Figura 8, diagrama UML amb les classes que conté el nostre programa.

Tant la classe Silhouette situada a l'esquerra del tot com la situada a la dreta del tot són la mateixa, la dupliquem només per no complicar la figura.

12.2. Explicació de les classes

12.2.1.ControladorHierarchy

Amb aquesta classe volem fer tots els càlculs necessaris de l'algorisme Hierarchy. L'algorisme el cridem de la llibreria scipy.cluster.

Les funcions que conté són:

- **retCluster(k, div):** Aquesta funció rep dos paràmetres: *k* que és el clúster que volem tornar i *div* que és el nombre de clústers en què hem dividit el data set.
  La funció retornarà una matriu amb tots els elements del clúster *k* havent dividit el data set en *div* clústers.
- **retLabels(k):** Aquesta funció rep un sol paràmetre *k*, que és el nombre de clústers en què dividim el data set.
  La funció retorna un vector que conté les etiquetes on s'indica a quin clúster pertany cada dada.
- **distanciaKClusters(k):** Aquesta funció rep un sol paràmetre *k*, que és el nombre de clústers en què dividim el data set.
  La funció retorna la suma de totes les distancies euclidianes dins dels clústers, dividint les dades en *k* clústers.
- **retAllClusters(k):** Aquesta funció rep un sol paràmetre *k*, que és el nombre de clústers en què dividim el data set.
  La funció retorna un vector amb *k* matrius, on cada matriu és un dels clústers en què ha dividit l'algorisme el data set.

12.2.2. ControladorKmeans

Amb aquesta classe volem fer tots els càlculs necessaris de l'algorisme Kmeans. L'algorisme el cridem des de la llibreria sklearn.cluster.

Les funcions que conté són:

- **executaKmeans(k):** Aquesta funció rep un sol paràmetre *k*, que és el nombre de clústers en què dividim el data set.
  Retrona l'execució de l'algorisme Kmeans dividint les dades en *k* grups.

- **tornaCentres(kmeans):** Aquesta funció rep un sol paràmetre *kmeans*, que és el resultat d'una execució de l'algorisme.
  Retorna un vector de tots els centres de l'execució de l'algorisme.
- **clusterPertany(kmeans, punt):** Aquesta funció rep una execució de *kmeans* i un punt del data set.
  Retorna l'índex del clúster al qual pertany el *punt*.
- **sumDistancies(kmeans):** Aquesta funció rep un sol paràmetre *kmeans*, que és el resultat d'una execució de l'algorisme.
  Retorna la suma de les distancies dels punts al centre del seu clúster.
- **returnLabels(kmeans):** Aquesta funció rep un sol paràmetre *kmeans*, que és el resultat d'una execució de l'algorisme.
  Retorna un vector amb índexs que marca a quin grup pertany cada dada.

12.2.3.AvaluadorHierarchyElbow

Aquesta classe ens serveix per a triar el nombre ideal de clústers utilitzant l'*elbow method*. En aquesta classe només utilitzarem l'algorisme Hierarchy.

Les funcions que conté són:

- **returnLabels():** Aquesta funció retorna un vector amb els índexs que ens marca a quin grup pertany cada dada un cop ja hem triat el nombre ideal de clústers.
- **trobaPendent():** Aquesta funció ens serveix per a calcular la pendent que hi ha en la gràfica de suma de distancies per nombre de clústers.
- **trobarNumClusters():** Aquesta funció retorna el nombre de clústers òptims amb els quals dividir el data set segons l'*elbow method*.

12.2.4.AvaluadorKmeansElbow

Aquesta classe ens serveix per a triar el nombre ideal de clústers utilitzant l'*elbow method*. En aquesta classe només utilitzarem l'algorisme Kmeans.

Les funcions que conté són:

- **returnLabels():** Aquesta funció retorna un vector amb els índexs que ens marca a quin grup pertany cada dada un cop ja hem triat el nombre ideal de clústers.
- **trobaPendent():** Aquesta funció ens serveix per a calcular la pendent que hi ha en la gràfica en la gràfica de suma de distancies per nombre de clústers.
- **trobarNumClusters():** Aquesta funció retorna el nombre de clústers òptims amb els quals dividir el data set segons l'*elbow method*.
- **returnWithLabels():**Aquesta funció retorna el nombre de clústers òptims amb els quals dividir el data set segons l'*elbow method*, i una matriu amb el data set original afegint-hi una columna amb l'índexs del clúster al qual pertany cada dada.

12.2.5.Silhouette

Aquesta classe ens serveix per a calcular el valor de silhouette d'un clustering i per a triar el nombre òptim de clústers dels algorismes. Per a el càlcul del valor silhouette hem emprat la llibreria *sklearn.metrics.silhouette_score*.

Les funcions que conté són:

- **retCenter(clúster):** Aquesta funció rep una matriu que conté les dades d'un mateix clúster.
  Amb aquesta funció calculem el centre del clúster i retornem el vector d'aquest punt.
- **retListClusters(k,labels):** Aquesta funció rep dos paràmetres, el valor de *k* és el nombre de dades en què està dividit un clúster. *Labels* és un vector amb els índexs que ens marca a quin és el clúster que pertany cada dada.

Retorna un vector de parelles de longitud *k.* La primera part de la parella conté una matriu amb tots els elements del clúster, i la segona part conté el vector del centre del clúster.

- **retClustering(k):** Aquesta funció rep un sol paràmetre *k*, que és el nombre de clústers en què dividim el data set.
  Retorna l'execució de l'algorisme dividint les dades en *k* clústers.
- **retLabels(k):** Aquesta funció rep un sol paràmetre *k*, que és el nombre de clústers en què dividim el data set.
  Retorna un vector amb els índexs dels clústers al qual pertany cada dada.
- **retPuntuacio(k):** Aquesta funció rep un sol paràmetre *k*, que és el nombre de clústers en què dividim el data set.
  Retorna la puntuació de l'índex *silhouette* dividint les dades en *k* grups.
- **buscaParcialClusters():** Amb aquesta funció volem saber quin és el nombre de clústers ideal segons el valor del *silhouette*. Retorna el nombre de clústers ideal segons *silhouette*.
- **analitza Final():** Aquesta funció és utilitzada per l'anterior per a trobar el punt ideal segons els valors de *silhouette*. Retorna un valor booleà segons si s'ha trobat el punt o no.

12.2.6.Davies-Bouldin

Aquesta classe ens serveix per a calcular el valor de l'índex *Davies-Bouldin* d'un clustering.

Aquesta classe només té una funció que és:

- **daviesbouldin(list,centres):** Aquesta funció rep dos paràmetres, el paràmetre *list* és un vector de matrius, on cada posició és la matriu d'un clúster. El paràmetre *centres* és una llista amb la posició de tots els centres.

12.2.7.Dunn

Aquesta classe ens serveix per a calcular el valor del *Dunn Index* d'un clustering.

Aquesta classe només conté una funció que és:

- **dunnindex(points,labels):** Aquesta funció rep dos paràmetres, *points* que és la matriu amb totes les dades del data set, i *labels* vector del clúster que pertany cada dada.

12.2.8.Avaluadors de bondat

Aquesta classe ens serveix de pont per a que l'avaluador general pugi obtenir els valors dels índexs i triar quin dels agrupaments és el millor.

12.2.9.Controlador general

Aquesta classe serveix com a pont entre l'*avaluador general* i les altres classes. Ens permet agilitzar els càlculs i evitar repetir càlculs. També ens permet que la classe *avaluador general* sigui més senzilla.

Les funcions que conté aquesta classe són:

- **trobaNumClusters():** Aquesta funció executa tots els avaluadors del nombre de clústers amb tots els algorismes, i guarda quins són aquest nombre de clústers.
- **selectBestKmeans():** Aquesta funció calcula el valor dels tres avaluadors de bondat per a totes les execucions de l'algorisme *Kmeans*

que tenen com a valor *k* un nombre entre el trobat per l'*elbowmethod* i l'índex *silhouette*.

Retorna una matriu on en cada fila contindrà els valors dels avaluadors de bondat de cada execució que està en el rang esmentat anteriorment.

- **selectBestHierarchy():** Aquesta funció calcula el valor dels tres avaluadors de bondat per a totes les execucions de l'algorisme *Hierarchy* divideix les dades en un nombre entre el trobat per l'*elbowmethod* i l'índex *silhouette*.

  Retorna una matriu on en cada fila contindrà els valors dels avaluadors de bondat de cada execució que està en el rang esmentat anteriorment.
- **selectBestClustering():** Aquesta funció utilitza tots els agrupaments que es calculen en les dues funcions anteriors, i els seus valors dels avaluadors de bondat, per a valorar quin de tots els agrupaments és el millor.

  Retorna l'algorisme que millor agrupa i el nombre de clústers ideal.

12.2.10. Avaluador general

Aquesta classe llegeix el document amb les dades que es volen estudiar i, automàticament crida al controlador general per al càlcul del millor algorisme amb el millor nombre de clústers possible.

12.3. Pseudocodi elbowmethod

En aquest apartat volem explicar com hem implementat l'algorisme que implementa l'elbowmethod. Per a implementar-ho ens hem basat en aquest pseudocodi:

```
elbow (min, max, obj)
    if (min == max) return min
    mid = (min + max)/2
```

```
pend = (f(mid)- f(mid+1))
if (pend > obj)
        elbow(mid, max, obj)
else
        elbow(min, mid, obj)
```

El cost computacional d'aquest algorisme és: $\log k$. On *k* és el nombre màxim de clústers considerats, en el nostre cas $\sqrt{n}$.

12.4. Pseudocodi silhouette

En aquest apartat volem explicar com hem implementat l'algorisme que implementa l'índex *silhouette*. Per a implementar-ho ens hem basat en aquest pseudocodi:

```
silhouette(min, max)
        i = min
        j = 1
        values = [ ]
        clusters = [ ]
        while i <= max :
                values.append(silhouette(min + i))
                clusters.append(i)
                i = i+j
                j = j*2
        pos = key(max(values))
        if (values[pos -1] >= values[pos+1]) :
                silhouette(clusters[pos-1],clusters[pos])
        else :
                silhouette(clusters[pos],clusters[pos+1])
```

El cost computacional d'aquest algorisme és: $\log k$. On *k* és el nombre màxim de clústers considerats, en el nostre cas $\sqrt{n}$.

13. Problemes amb el software

Alhora de desenvolupar el software, ens hem trobat amb diversos problemes. Volem destacar un problema que ha estat el més important, i explicarem com l'hem solucionat.

Durant el desenvolupament del programa ens vàrem donar compte que els temps d'execució eren molt elevats. Per veure on el programa trigava més vam fer un estudi de temps dels algorismes.

Per a trobar aquestes perdudes de temps hem utilitzat un data set que està en la llibreria de sklearn.datasets. Aquest data set té 1797 files i 64 columnes.

Amb aquest data set vam veure que els temps de cada algorisme eren:

Kmeans: 7,34 sec
Hierarchy: 235,7 sec

Com podem veure en els temps d'execució dels algorismes, el Hierarchy produeix que el temps d'execució del programa sigui molt elevat. Això és degut al cost computacional dels algorismes. El cost de l'algorisme Hierarchy és: $O(n^2 \log n)$, sent *n* el nombre de punts del data set d'entrada. Per altre banda el cost computacional del Kmeans és: $O(nkdi)$ on *n* és el tamany de l'entrada de dades, *k* el nombre de grups en què dividir les dades, *d* el nombre de variables que tenen les dades i *i* el nombre d'iteracions que fa al algorisme.

Per evitar aquests costos de temps tan elevats hem decidit executar el programa solament amb l'algorisme Kmeans, i un cop hem tingut els resultats preguntar al usuari si vol que fem un estudi més extens de les dades.

D’aquesta manera el temps en què mostrem els primers resultats és redueixen significativament, i donem la possibilitat a que si l’usuari vol tenir un estudi més extens amb un cost temporal més elevat, el pugui fer.

14. Jocs de prova

Per a validar el programa hem provat aquest amb diversos data sets que porten dades mèdiques. Hem utilitzat tres data sets, dos dels quals ja han estat estudiats i un altre que no.

14.1. Joc de prova 1

En aquest joc de dades tenim informació de pacients que han sofert un ictus cerebral i han sobreviscut amb una edat superior o igual a 65 anys. El data set és d'una institució sanitària centrada en l'atenció a la gent gran, geriatria i envelliment, concretament el *Parc Sanitari Pere Virgili*.

Per aquest data set hi ha un treball d'anàlisis amb el que compararem els resultats que obtenim, del qual ens referirem constantment en el anàlisis dels resultats [8].

14.1.1. Anàlisi de les dades

Les dades contenen 384 pacients, dels quals tenim 9 atributs. Aquestes dades són:

- **CharIson:** Índex utilitzat en geriatria per a fer una valoració del estat del pacient.
- **Num Cuidador :** variable booleana que ens diu si el pacient té un cuidador a casa o no.
- **Edat:** variable que ens indica l'edat del pacient
- **Esatocivilnum:** variable booleana que ens indica si el pacient viu amb companyia o viu sol.

- **IB_post:** Índex utilitzat en geriatria per a fer una valoració del estat del pacient, el nom de l'índex és Barthel. Aquesta variable representa l'índex Barthel abans del ictus.
- **IB_pre:** Índex utilitzat en geriatria per a fer una valoració del estat del pacient, el nom de l'índex és Barthel. Aquesta variable representa l'índex Barthel després del ictus.
- **Nihss:** Índex utilitzat en geriatria per a fer una valoració del estat del pacient.
- **RanchoLosAmigosNum:** Índex utilitzat en geriatria per a fer una valoració del estat del pacient, el nom ve donat d'un centre geriàtric de Califòrnia.
- **milloraIB:** Índex utilitzat en geriatria per a fer una valoració del estat del pacient, el nom de l'índex és Barthel. Aquesta variable representa la millora de l'índex Barthel en el moment posterior al ictus i 6 mesos després del ictus.

Aquestes dades només contenen 3 valors mancants, 2 dels quals són de variables booleanes i les hem canviat per 0. L'ultima és l'índex RanchoLosAmigosNum, i hem agafat un valor mig de totes les observacions.

14.1.2.Mètodes de selecció del nombre de clústers 1

Inicialment mostrarem els resultats obtingut utilitzant solament l'algorisme *kmeans*. Inicialment definim el rang de possibles *k* entre 2 i $\sqrt{384} = 19$.

Començarem fent l'estudi del nombre de clústers ideal amb l'*elbowmethod*.

Ara calculem la suma de les distancies euclidianes a les solucions de l'execució de l'algorisme *kmeans* amb els *k* extrems del rang inicial. Els resultats són respectivament: 505044 i 89625, amb els que podem calcular el pendent objectiu que és: -23608.

Aplicant la cerca dicotòmica, hem calculat els següents valors de suma de distàncies:

Número Clústers	Suma de distancies
2	505044
4	285489
5	228578
6	199072
7	179647
19	89625

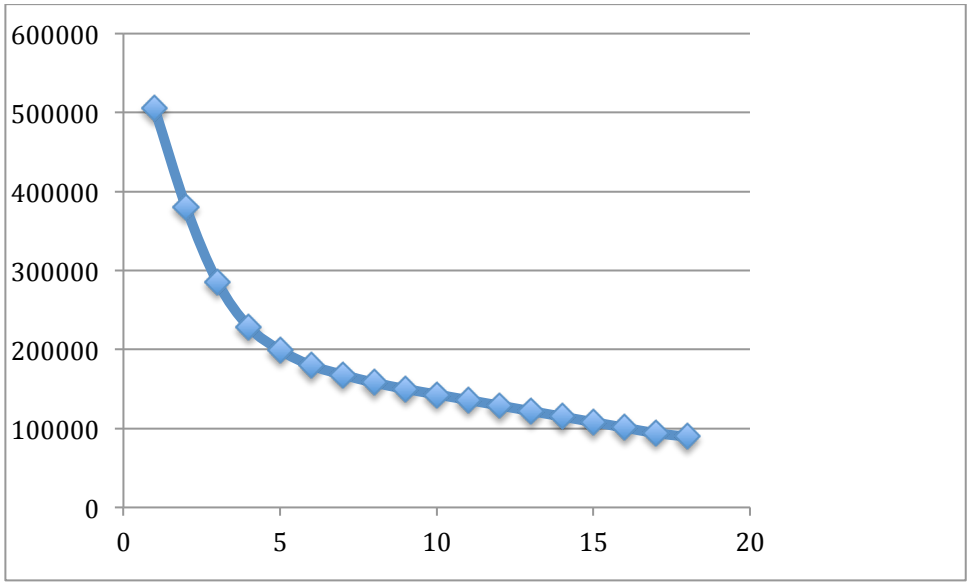

Figura 9, gràfic que relaciona el nombre de clústers amb la suma de distancies euclidianes.

Com podem veure a la Figura 9 i segons el mètode, el nombre ideal de clúster seria 4 o 5. Amb el nostre algorisme, el pendent que hi ha entre 5 i 6 clústers és -29505, comparant amb el pendent entre 4 i 5 clústers (-56911) i el pendent entre 6 i 7 clústers (-19424), la que més s'aproxima és aquesta. Per tant aquest mètode selecciona com a millor nombre de clústers 5.

Ara farem el estudi amb el mètode *silhouette*. Amb aquest mètode hem calculat els següents índexs:

Número Clústers	Valor de l'índex
2	0.3015
3	0.3034
4	0.3255
5	0.3403
6	0.3253
9	0.2821
17	0.2538

Veient els resultats, veiem que el nombre de clústers que obté és el de 5.

Com que els dos mètodes han seleccionat com a millor nombre de clústers el mateix, no cal aplicar els índexs d'avaluació. El millor nombre de clústers és 5.

Tot i això els valors obtinguts dels valors són:

Dunn	Silhouette	Davies-Bouldin
0.0815	0.3403	0.9400

14.1.3.Mètodes de selecció del nombre de clústers 2

En aquest cas farem el mateix estudi que en el punt anterior, però en aquest cas hem decidit eliminar la columna *milloraIB*, ja que és una dada que creiem que no impacte en els resultats.

Amb aquest data set obtenim els següents valors amb l'*elbow method*:

Número Clústers	Suma de distancies
2	248232

4	113370
5	92383
6	81553
7	73484
10	57197
11	54087
19	38163

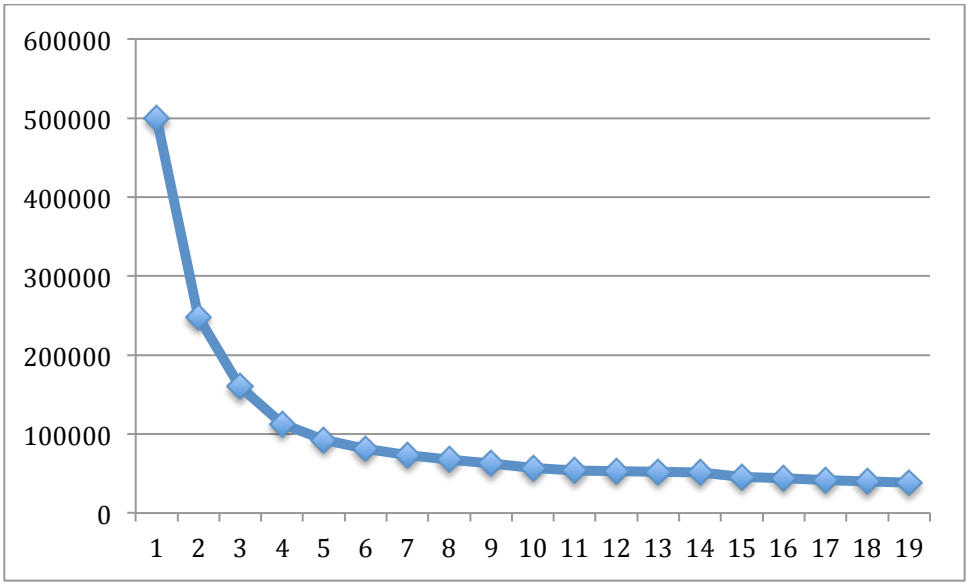

Figura 10, gràfic que relaciona el nombre de clústers amb la suma de distancies euclidianes.

Amb el nostre algorisme el resultat que obtenim és que el nombre de clústers ideal és 4.

Ara fem el estudi amb el mètode *silhouette*, amb el que obtenim els següents valors:

Número de Clústers	**Valor de l'índex**
2	0.4205
3	0.4493

4	0.3673
5	0.3531
9	0.2810
17	0.2509

Amb aquest resultats que obtenim de l'índex, el nostre algorisme selecciona com a millor nombre de clústers 3.

Com que en els dos algorismes no hem obtingut el mateix nombre de clústers, hem d'emprar els índexs d'avaluació. Els resultats que obtenim són:

Num Clústers	**Dunn**	**Silhouette**	**Davies Bouldin**
3	0.0772	0.4493	0.7991
4	0.044	0.3673	0.8887

Amb aquest resultats i aplicant la formula per la tria del millor clustering, el nombre de clústers ideal és 3.

Ens adonem que amb aquest mètode s'aconsegueixen uns valors millors que els que havíem obtingut en el experiment anterior. És per això que donem validesa a la hipòtesis que la variable *milloraIB,* no dona informació rellevant alhora d'aplicar clustering.

14.1.4.Mètodes de selecció del nombre de clústers 3

Ara, amb el mateix joc de dades també utilitzarem l'algorisme aglomeratiu també, tampoc utilitzarem la columna amb la informació sobre la dada *milloraIB*. Les dades que obtenim amb l'algorisme *kmeans* són els mateixos que en els experiments anteriors.

Utilitzant el mètode *Hierarchy* i l'*elbowmethod* obtenim els següents valors:

Número Clústers	Valor de l'índex
2	1425629
4	619440
5	377815
6	301779
7	266888
10	127203
11	113277
19	65748

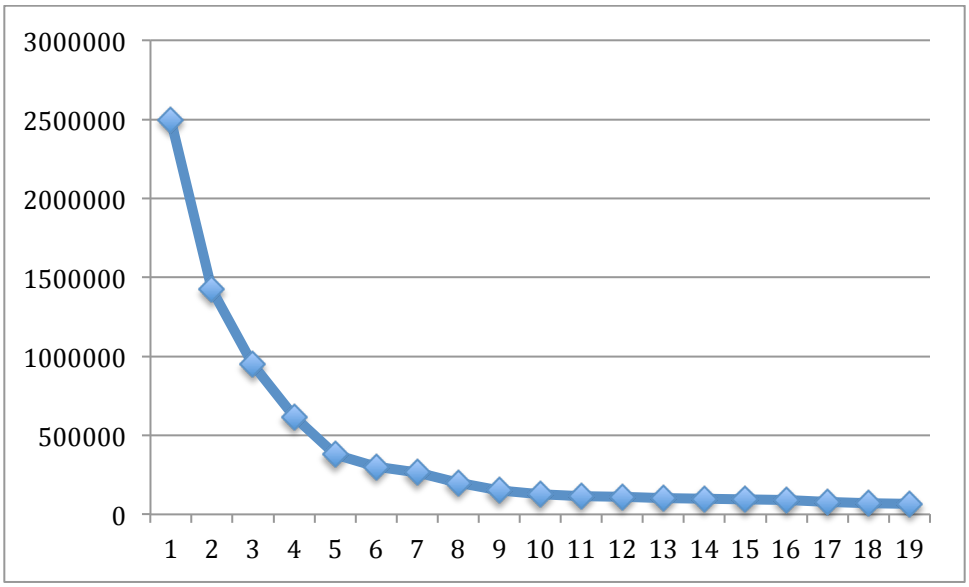

Figura 11, gràfic que relaciona el nombre de clústers amb la suma de distancies euclidianes.

L'algorisme que utilitzem per implementar l'*elbowmethod* selecciona com a nombre de clústers 5. Com podem veure en la figura 11, el colze es troba en aquest punt.

La taula amb les puntuacions que obtenim amb e l'índex *silhouette* són:

Número Clústers	**Valor de l'índex**
2	0.4072
3	0.3567
5	0.3567
9	0.25816
17	0.2343

Com es pot veure en la taula, el millor índex s'obté amb nombre de clústers 2, que coincideix amb el nombre de clústers que selecciona l'algorisme.

Ara ja tenim delimitat els nombre de centres que hi ha amb l'algorisme *Hierarchy*, que són 2 i 5. Ara calculem tots els valors dels índexs avaluadors, que juntament amb els de l'algorisme *kmeans* obtenim aquesta taula:

Algorisme	**Num Clústers**	**Dunn**	**Silhouette**	**Davies Bouldin**
Kmeans	3	0.0772	0.4493	0.7991
Kmeans	4	0.044	0.3673	0.8887
Hierarchy	2	0.0534	0.4072	0.8304
Hierarchy	3	0.0617	0.3567	1.013
Hierarchy	4	0.0833	0.3301	1.0388
Hierarchy	5	0.0817	0.3021	1.0515

Aplicant la formula que tenim per a seleccionar el millor algorisme, el resultat que obtenim és que el millor algorisme és el *kmeans* i el millor nombre de clústers 3.

14.1.5.Conclusions

Si analitzem els resultats que s'obtenen en el treball que es varen fer amb les dades, ells arriben a la conclusió que per l'algorisme *kmeans* el millor nombre de clústers és 3, tot i que dividint les dades en 5 clústers també el trobaven bo. El nostre treball ha arribat a estudiar un rang entre 2 i 5 clústers, seleccionant com a millor el clustering el mateix que és va utilitzar. A partir de dividir les dades ells arriben a conclusions molt interesants, inclús arriben a fer un arbre de decisió per a la tria de a quin clúster pertany cada dada amb un alt nombre d'encerts.

A unes de les conclusions que podem arribar és que les variables que no aportin clarament informació a les dades, són millor descartar-les. Això ho podem dir veient els resultats que hem tingut quan hem utilitzat la variable *milloraIB* i sense utilitzar-la. Comparant aquest resultats amb el estudi previ, veiem que sinó utilitzem la dada obtenim uns resultats semblants. També els valors obtinguts per els índexs avaluadors són millors.

14.2. Joc de prova 2

En aquest joc de proves estudiarem les dades de pacient que varen ingressar en l'Hospital de Sant Pau de Barcelona. En les dades tenim uns índexs que ens indiquen quins símptomes té cada pacient. No disposem un treball previ amb aquestes dades.

14.2.1.Anàlisi de les dades

En aquest joc de dades disposem de 21043 files de dades amb 57 columnes d'informació.

Al analitzar les dades ens vàrem adonar que hi havia dos tipus de pacients, uns que tenien definit un *id*, i uns altres que no. Els pacients sense *id* són un total

de 262, i són els únics que contenen informació d'unes certes columnes. Això ens ha fet pensar que devien ser estrangers sense identificació i és per això que teníem més informació com ara el país o la data de naixement. Per evitar problemes i veient que el nombre de pacients com aquests són pocs, hem decidit descartar tots aquests pacients.

La informació dels pacients que tenim i que considerem útil per a l'analisis són:

- **D_ingres:** Dia en què el pacient va ingressar.
- **D_alta:** Dia en què el pacient va obtenir l'alta mèdica.
- **C_ingres:** Condicions amb les quals el pacient va ingressar. En general si va ser urgent o programada.
- **C_alta:** Condicions amb les que el pacient va abandonar el hospital. En general si va anar a un altre centre o va tornar cap a casa.
- **Sexe**
- **D_naix:** Data de naixement.
- **DP:** Diagnòstic principal.
- **DS1...DS9:** Altres símptomes que es varen diagnosticar en el ingrés del pacient.
- **CIP:** el ID del pacient.

Els Diagnòstics venen donats per una nomenclatura que es diu: "International Statistical Classification of Diseases and Related Health Problems" [14].

Per a fer el anàlisi de clustering, hem aplicat un prepocés sobre les dades. Primer hem passat tota la informació dels diagnòstics dels pacients a una matriu booleana, indicant amb un 1 en la posició mat[i][j] si el pacient *i* té el símptoma *j*. També fusionem els pacients amb les mateixes ID, ja que es tracte de la mateixa persona.

Un cop tenim la matriu booleana de tots els diagnòstics dels pacients fusionada, fem un anàlisi de totes les columnes, i eliminem aquelles que menys d'un 2% dels pacients tenen. Així evitem tenir informació que sembla poc rellevant.

Per tractar els missings que trobem en el data set, hem decidit canviar les variables que ens falten per la mitjana de la columna.

Al final ens queda una matriu de dimensions 21044 per 58 columnes, que produirà que els temps d'execució siguin molt elevats.

14.2.2.Mètodes de selecció del nombre de clústers

El rang inicial del nombre de clústers és entre 2 i 145 = $\sqrt{21044}$.

Aplicant el mètode *elbowmethod* per a la selecció de la millor *k* obtenim els següents valors de suma de distancies euclidianes:

Número Clústers	Valor de l'índex
2	69874
73	43072
74	42848
78	42535
79	42305
80	42387
81	42231
82	42040
83	42063
91	41509
92	41408
109	40214
110	40076
145	38348

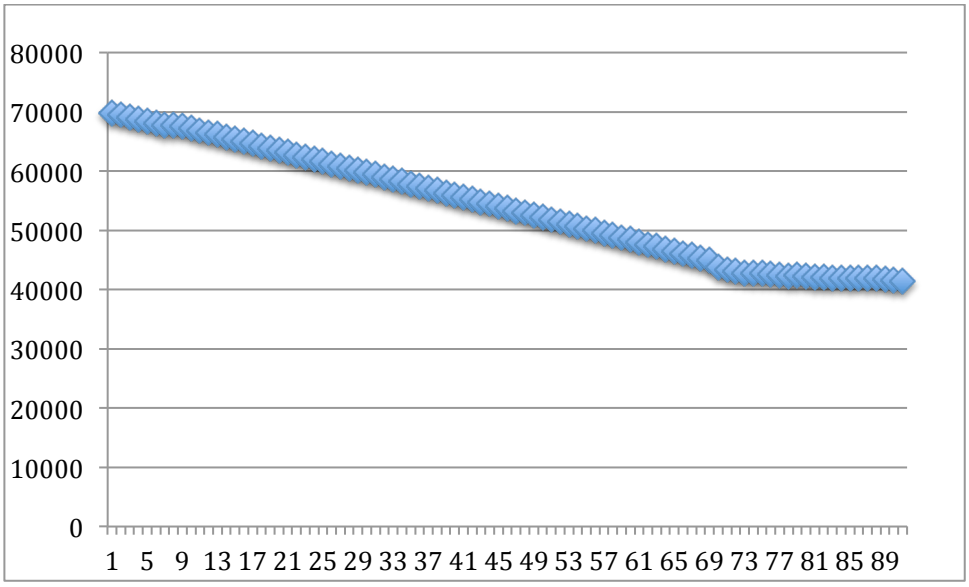

Figura 12, gràfic que relaciona el nombre de clústers amb la suma de distancies euclidianes.

Tal i com podem veure a la taula, el augment d'un clúster mai implica una millora molt gran en la suma de les distancies euclidianes. Per aquest motiu, el gràfic de la Figura 12 surt una representació dels punts en forma de línia. Amb aquest resultats, podem dir que l'*elbow method* no s'adapta bé amb aquest data set.

El nombre ideal de clúster obtingut per aquest mètode és 78.

La taula amb les puntuacions que obtenim amb l'índex *silhouette* són:

Número Clústers	**Valor de l'índex**
2	0.0760
3	0.0689
5	0.0618
9	0.0642
17	0.0636
33	0.0656
129	0.0744

Com podem veure en els resultats de la taula, tots els valors obtinguts per l'índex són molt baixos. Aquests resultats ens índica que els clústers aconseguits amb l'algorisme no aconsegueix clústers molt bons.

El nombre de clústers amb un millor valor de *silhouette* és 2. Per tant els dos extrems elegits són 2 i 78 clústers.

El problema que trobem en aquest data set és el gran nombre d'atributs (58). Això crea un problema anomenat "curse of dimensionality"[17], és un fenòmen ben conegut que crea molts problemes als algorismes de clustering basats en la distància. Amb un nombre de variables tant gran, la distància entre parelles de punts és molt gran. També el fet que les dades siguin heterogènies, fa que el procés de clustering sigui complicat.

Com el mètode *Hierarchy* també té en compte la distància, no trobaríem una gran millora en els resultats obtinguts. Per evitar els temps d'execució elevats del algorisme, no hem trobat convenient aplicar aquest algorisme.

14.2.3.Conclusions

La conclusió principal que podem treure d'aquest joc de proves és que el nostre programa no funciona correctament amb qualsevol joc de dades. En aquest cas, el joc de dades és molt ampli i les variables que conté són molt diferents unes amb altres. Segurament si haguéssim definit la població del joc de dades segons uns criteris, com per exemple que tinguin un o varis símptomes o bé una edat definida haguéssim obtinguts uns millors resultats.

Amb aquest joc de proves podem veure que si el joc de dades no està ben tractat no obtenim bons resultats. Per tant hem de tenir clar que el joc de dades ha de ser seleccionat correctament, amb una població amb la qual es puguin treure conclusions i que aquest data set necessita d'un bon preprocés.

14.3. Joc de prova 3

En aquest joc de proves disposem del data on les dades són demogràfiques de gent extretes per Barry Becker del 1994 Census data base[13]. Aquestes dades contenen una dada diferencial que ens diu si la persona cobra més de 50 mil dòlars o no.

Aquest data set conté 65122 observacions amb variables categòriques i numèriques, fent un total de 14 variables. El data set és del 01-05-1996, i conté missing vàlues.

El treball que utilitza el data set tenia com a objectiu per classificar. En el nostre cas les utilitzarem solament per el anàlisis de clustering.

14.3.1.Anàlisi de les dades

En el preprocés d'aquest data set el que fem és assignar valors a les variables categòriques.

Les variables que contenen aquest data set són:

- **Age:** Variable numèrica que representa l'edat de la persona.
- **Worckclass:** Variable categòrica que defineix el tipus de classe de la persona.
- **Education:** Variable categòrica que defineix el tipus d'educació de la persona.
- **Marital-status:** Variable categòrica que defineix quin és el estat sentimental de la persona.
- **Occupation:** Variable categòrica que defineix el tipus de treball de la persona.
- **Race:** Variable categòrica que defineix la raça de la persona.
- **Sex:** Variable categòrica que defineix el sexe de la persona.
- **Capital-gain:** Variable numèrica que indica els guanys de la persona.

- **Capital-loss:** Variable numèrica que indica les pèrdues de la persona.
- **Hours per week:** Variable numèrica que ens indica la quantita d'hores que treballa la persona.
- **Native Country:** Variable categòrica que ens indica el país natiu de la persona.

Per a tractar els errors en el joc de dades, hem substituït aquests valors per la mitjana de tota la columna. En 5 files ens hem trobat que faltava un dels valors inicials (l'allargada de la fila era més petit), com són pocs casos els hem descartat.

Al executar aquest joc de dades, al contenir moltes columnes quan hem calculat els valors de *silhouette* hem tingut un error de memòria. Per evitar aquests errors i fer que l'execució del programa sigui més ràpida, hem dividit les dades en 10 matrius del mateix tamany. L'assignació de cada observació a un dels grups és aleatori.

Segurament perdrem una mica de precisió alhora de calcular els valors dels índexs avaluadors, però no esperem que siguin molt significatius aquests errors.

14.3.2.Mètodes de selecció del nombre de clústers 1

En aquest apartat només mostrarem els resultats dels valors obtinguts per els mètodes de selecció del nombre de clústers que obtenim en el primer grup. Després mostrarem per a cada grup l'algorisme i amb quantes divisions fa la selecció el nostre algorisme.

Ara cada grup tindrà 6512 variables, per tant els rangs del nombre de clústers estaran entre 2 i 80.

En aquests resultats solament si apliquem l'algorisme *kmeans* :

Aplicant el mètode *elbowmethod* per a la selecció de la millor *k* obtenim els següents valors de suma de distancies euclidianes:

Número Clústers	Valor de l'índex
2	62239588621
6	6953150891
7	5066798560
8	3869443671
9	3076189412
10	2441855043
11	2067472082
12	1727363820
21	565428953
22	517150287
41	151000449
42	143436362
80	42425680

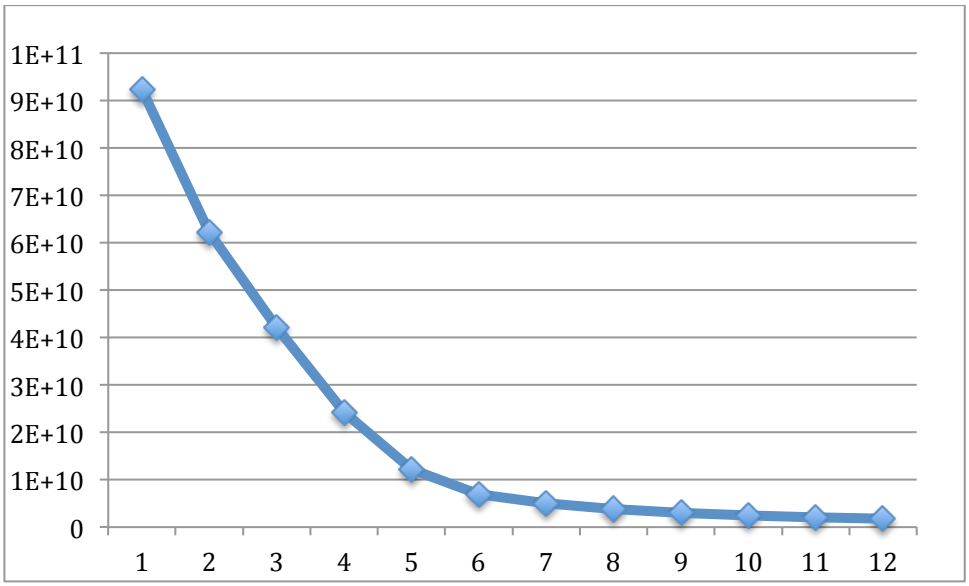

Figura 13, gràfic que relaciona el nombre de clústers amb la suma de distancies

euclidianes.

En el gràfic de la figura 13, amb el que sol hem representat els 12 primers valors ja que els altres valors eren molt propers al eix horitzontal, podem veure com el colze podria ser *k* amb valor 5 o amb valor 6.
El nostre programa ens selecciona com a millor nombre de clústers 8. Això és degut a que busca el punt amb el mateix pendent que hi entre la suma euclidiana amb dos clústers i la suma entre 80, cosa que provoca que aquesta pendent sigui més proper a 0 que el pendent que hi ha en el colze. Si haguéssim definit el rang inicial entre 2 i 12, el nostre algorisme hagués triat com a millor *k* el valor 5. Tot i així el valor que ha obtingut el programa no està molt lluny del òptim.

La taula amb les puntuacions que obtenim amb l'índex *silhouette* són:

Número Clústers	**Valor de l'índex**
2	0.6264
3	0.5907
5	0.5655
9	0.5458
17	0.5320
33	0.5216
65	0.5320

Com veiem a la taula, el nombre de clústers amb valor 2 obté el millor resultat. L'algorisme selecciona com a millor nombre *k* utilitzant e l'índex *silhouette* el valor 2.

Ara apliquem els índexs avaluadors per a saber quin clustering és el millor per a totes aquelles *k* entre 2 i 8 incloses.

Els resultats són:

Num Clústers	Dunn	Silhouette	Davies-Bouldin
2	0.0014	0.62	0.50
3	0.0017	0.59	0.50
4	0.0017	0.57	0.50
5	0.0035	0.56	0.50
6	0.0023	0.55	0.50
7	0.0025	0.54	0.50
8	0.0038	0.54	0.50

Si analitzem la taula i la comparem amb els resultats obtinguts en el joc de proves 1, veiem que l'índex de *Dunn* té uns valors més dolents. En canvi el valor del *Silhouete* i de l'índex del *Davies-Bouildin* són millors. Comparant els resultats veiem que en el joc de proves 1 s'agrupaven millor les dades segons el nostre algorisme. Tot i així no podem dir amb exactitud quina de les dues bases de dades és més fàcil d'agrupar.

Per els valors obtinguts en la taula anterior el nostre algorisme ha seleccionat com a millor clustering, aquell que divideix les dades en 8 grups.

14.3.3.Mètodes de selecció del nombre de clústers 2

Ara mostrarem els resultats que obtenim si també apliquem l'algorisme aglomeratiu. Primer mostrarem els resultats que obtenim alhora de calcular la millor *k* aplicant solament l'algorisme.

Aplicant el mètode *elbowmethod* per a la selecció de la millor *k* obtenim els següents valors de suma de distancies euclidianes:

Número Clústers	Valor de l'índex

2	45097700386
6	4612145063
7	3802083937
8	2718709579
9	2094224411
10	1687829783
11	1485639248
12	1228741030
21	386593892
22	362633871
41	109789428
42	105687063
80	30667195

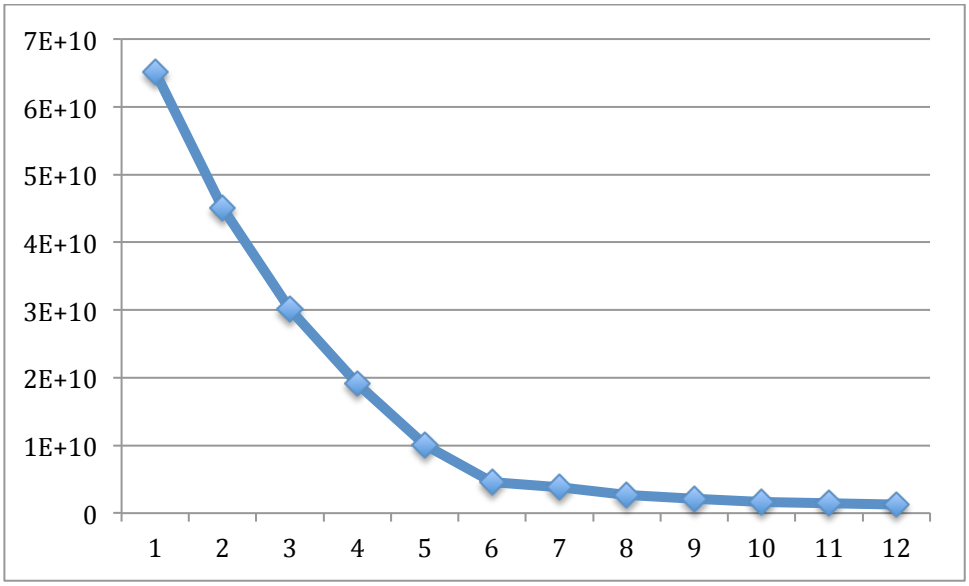

Figura 14, gràfic que relaciona el nombre de clústers amb la suma de distancies euclidianes.

Com passava en l'algorisme *kmeans,* aquest mètode ens ha seleccionat coma millor *k* el valor 8, tot i que no està exactament en el colze. Com hem comentat prèviament això és degut a que el rang màxim de nombre de clústers és molt alt, i això produeix que el pendent que busquem sigui més proper a 0.

La taula amb les puntuacions que obtenim amb l'índex *silhouette* són:

Número Clústers	Valor de l'índex
2	0.6076
3	0.5694
5	0.5246
9	0.5224
17	0.4849
33	0.4809
65	0.4650

Com veiem a la taula, el nombre de clústers amb valor 2 obté el millor resultat. L'algorisme selecciona com a millor nombre *k* utilitzant l'índex *silhouette* el valor 2.

Ara apliquem els índexs avaluadors per a saber quin clustering és el millor per a totes aquelles *k* entre 2 i 8 incloses.

Els resultats són:

Num Clústers	Dunn	Silhouette	Davies-Bouldin
2	0.0021	0.60	0.49
3	0.0021	0.56	0.50
4	0.0031	0.56	0.49
5	0.0033	0.52	0.50
6	0.0036	0.53	0.50
7	0.0046	0.51	0.50
8	0.0032	0.51	0.50

Com podem veure en els resultats de la taula anterior i comparant-los amb els que hem obtingut amb l'algorisme *kmeans*, els valors dels dos algorismes són molt semblants. Tot i això, l'algorisme que obté millors puntuacions és el *Hierarchy* amb 7 grups.

En la següent taula mostrarem quin algorisme i amb quants grups és el seleccionat per el nostre algorisme depenent de quin dels 10 grups de dades triem.

Numero del grup	Algorisme	Nombre de clústers
1	Hierarchy	7
2	Kmeans	7
3	Hierarchy	8
4	K-means	8
5	Hierarchy	6
6	Hierarchy	7
7	K-means	8
8	Hierarchy	8
9	K-means	6
10	Hierarchy	6

Veient els resultats ens adonem que no tots seleccionen el mateix algorisme ni el mateix nombre de clústers. El algorisme K-means és seleccionat un total de 4 cops per 6 cops del Hierarchy. Com el nombre de repeticions de cada algorisme és molt similar, creiem que el més oportú és seleccionar el K-means perquè el seu cost computacional és més petit. El nombre de clúster que selecciona el algorisme vaira entre 6 i 8 de forma equilibrada. Com no podem garantir la bondat de un nombre de clústers respecte un altre, creiem que qualsevol dels tres és suficientment bo.

14.3.4.Conclusions

Tot i que havent dividit les dades hem perdut validesa dels resultats, veiem que tots els grups seleccionen un nombre de clústers dins d'un rang petit. Tot i això partint les dades guanyem en temps d'execució. Per tant, creiem que el fet de partir les dades no és dolent sempre i quan es tingui present que els resultats poden variar una mica.

En aquest joc de dades ens adonem que tant l'algorisme *K-means* com el *Hierarchy* agrupen bé les dades, però per rapidesa d'execució ens quedaríem amb l'execució del primer.

Com veiem en el cas que solament analitzem el primer grup de dades, el nombre de clústers que selecciona l'*elbow method* està una mica desplaçat respecte el ideal, però gracies a que e l'índex *silhouette* selecciona un altre nombre de clústers, el nombre de clústers seleccionat finalment s'adapta millor.

15. Conclusions

Un cop aplicat el nostre programa als tres jocs de proves ens hem adonat de diverses coses. En general podem dir que si les dades han estat ben seleccionades i preprocesades, els resultats que obtenim amb el programa són bons.

Primer de tot, hi ha dades més fàcils d'agrupar que altres. Com hem vist en el joc de proves 1, unes dades que es varen utilitzar en un treball amb un resultats satisfactoris, el nostre programa troba el mateix nombre de clústers amb el que es va treballar en el treball. Ara si el joc de dades conté observacions que no mostren una gran diferencia entre unes i altres, o no conté variables que les categoritzi els resultats que obté el nostre programa no són gens satisfactoris.

Si analitzem els dos algorismes que utilitzem per a la selecció del nombre de clústers òptim, trèiem diverses conclusions. Si veiem els resultats que proposa l'índex *silhouette* són molt més petits que els que proposa l'*elbow method*, en general sempre són 2 o 3 clústers el que recomana. Això pot ser degut a que el *silhouette* penalitza que hi hagin més clústers, segurament si poguéssim ajustar aquesta penalització obtindríem uns resultats més diversos. La selecció del nombre de clústers que realitza el nostre *elbow method* en general és la correcte si analitzem les dades, però és cert que en alguns casos està una mica desplaçat aquest valor. El problema amb la cerca que fem del colze òptim és la dependència que hi ha amb el rang que definim. Per solucionar aquest problema es podria canviar els rangs inicials del nombre de clústers, d'aquesta forma el pendent que busca el nostre algorismes s'aproximarà més al pendent del colze.

Si analitzem els dos algorismes d'agrupament que hem utilitzat, veiem que tant un com l'altre obtenen resultats satisfactoris. Si haguéssim de triar un dels dos ens quedaríem amb el *kmeans*, el principal motiu és l'eficiència, *kmeans* és

més ràpid que l'algorisme aglomeratiu i en els casos amb conjunt de dades grans com és el nostre, el temps d'execució del aglomeratiu era molt superior a *k-means*.

Si analitzem els tres índexs que utilitzem per a elegir el millor clustering, veiem que en general els valors que obtenim per als diferents clustering són semblants, cosa que ja esperàvem perquè només són calculats en aquells clusterings que ja havíem preseleccionat abans com a candidats. En general hem vist que els valors obtingut en l'avaluació són bons.

Si utilitzem un joc de dades amb un gran nombre de dades, com és en el cas del tercer joc de proves, recomanem dividir el data sets de forma aleatòria i en grups del mateix tamany. D'aquesta forma el programa s'executarà més ràpidament, encara que perdem una mica de precisió en la selecció del clustering, en el nostre cas el nombre de clúster té una variació de ± 1.

En general arribem a la conclusió que si fem una selecció bona de les dades i apliquem un bon preprocesament aconseguim uns bons resultats.

Com a treballs futurs es podrien estudiar millor alguns dels passos fets. Uns exemples serien: Un estudi de quins rangs són els apropiats alhora d'aplicar el nostre *elbowmethod*. Mirar de calibrar una mica l'índex del *silhouette* per que no penalitzi tant incrementar el nombre de clústers. Buscar un altre mètode amb el que valorar la bondat dels clusterings.

16. Bibliografia

[1] Rousseeuw, P.J.; Kaufman, L. (1990). Finding Groups in Data: An Introduction to Clúster Analysis. Wiley

[2] John McCullock. (2012). K-means Introduction. Recuperat de:
http://mnemstudio.org/clustering-k-means-introduction.htm

[3] Moya, R. (2016). Gràfic en forma de punts de les dades representat en 2D. [Figura]. Recuperat de
http://www.nbertagnolli.com/jekyll/update/2015/12/10/Elbow.html

[4] Raschka's, S. (2013). Diagrama de l'maielbow method. [Figura]. Recuperat de https://sebastianraschka.com/faq/docs/issues-with-clustering.html

[5] Wheley, R. (2014). Dendograma empleant l'lagorisme Hierarchy. [Figura]. Recuperat de https://stackoverflow.com/questions/16883412/how-do-i-get-the-subtrees-of-dendrogram-made-by-scipy-cluster-hierarchy

[6] Charnelli,M. (2017). Gràfic amb els índexos silhouette. [Figura]. Recuperat de:
https://www.researchgate.net/publication/317720184_Modeling_Students_through_Analysis_of_Social_Networks_Topics

[7] Grup d'investigació LARCA (2017). Laboratory for Realational Algorithmics, Complexity and Learning. Barcelona, ESP. Recuperat de:
http://recerca.upc.edu/larca/en

[8] Perez,L. (2016). Rehabilitation Profiles of Older Adult Stroke Survivors. Barcelona, ESP. Recuperat de: http://journals.plos.org/plosone/article/authors?id=10.1371/journal.pone.0166304

[9] Cárdenas, M. Indice de Davies Bouldin. Recuperat de: http://wwwae.ciemat.es/~cardenas/docs/lessons/Davies-Bouldin.pdf

[10] Wikipedia. (2016). Elbow method (clustering). Recuperat de: https://en.wikipedia.org/wiki/Elbow_method_(clustering)

[11] Wikipedia. (2017). Silhouette (clustering). Recuperat de: https://en.wikipedia.org/wiki/Silhouette_(clustering)

[12] Ramos, J. (2012). Dunn's Index. Recuperat de: https://es.mathworks.com/matlabcentral/fileexchange/27859-dunn-s-index

[13] Kohavi, R i Beckler, B. (2001). Adult data set. Silicon Graphics. Recuperat de: https://archive.ics.uci.edu/ml/datasets/adult

[14] Wikipedia. (2017). International Statistical Classification of Diseases and Related Health Problems. Recuperat de: https://en.wikipedia.org/wiki/International_Statistical_Classification_of_Diseases_and_Related_Health_Problems

[15] Bertagnolli, N. (2015). Elbow method and finding the rigth number of clústers. Recuperat de: http://www.nbertagnolli.com/jekyll/update/2015/12/10/Elbow.html

[16] Ruffini, M i Gavaldà, R. (2017). Clustering with Tensor Decomposition. Recuperat de: https://arxiv.org/pdf/1708.08994.pdf

[17] Wikipedia. (2017). Maldición de la dimensión. Recuperat de: https://es.wikipedia.org/wiki/Maldición_de_la_dimensión

17. Software

- Python. (2016). Python 2.7.12. Recuperat de: https://www.python.org/downloads/release/python-2712/

- Numpy. (2017). Numpy Package. Recuperat de: http://www.numpy.org

- Scipy. (2017). Scipy library. Recuperat de: https://www.scipy.org/scipylib/index.html

- Scikit learn. (2017). Sklearn library. Recuperat de: http://scikit-learn.org/stable/